ADAPTIVE TOOL SELECTION STRATEGIES FOR DRILLING IN FLEXIBLE MANUFACTURING SYSTEMS

A Thesis

by

KARTHIK CHANDER

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

May 2004

Major Subject: Industrial Engineering

ADAPTIVE TOOL SELECTION STRATEGIES FOR DRILLING IN FLEXIBLE MANUFACTURING SYSTEMS

A Thesis

by

KARTHIK CHANDER

Submitted to Texas A&M University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Approved as to style and content by:

Amarnath Banerjee (Chair of Committee)	Cesar O. Malave (Member)
Barry Lawrence (Member)	Mark L. Spearman (Head of Department)

May 2004

Major Subject: Industrial Engineering

ABSTRACT

Adaptive Tool Selection Strategies for Drilling in Flexible Manufacturing Systems.
(May 2004)
Karthik Chander, B.E., Bangalore University, Bangalore, India
Chair of Advisory Committee: Dr. Amarnath Banerjee

The thesis presents an approach to adaptive decision making strategies to reduce bottlenecks in a drilling operation and to extend tool life. It is an attempt to portray the real drilling system in a typical Flexible Manufacturing System (FMS) layout. The system designed serves as back end intelligence to drilling machines (*INTELLIDRILL*) in a Flexible Manufacturing System for making dynamic and real time decisions. *INTELLIDRILL* uses mathematical and adaptive tool reliability models to simulate the machining conditions and tool availability for an operation based on history of tool failures. The results are used to compute the machining parameters and the tools required for an operation. *INTELLIDRILL* can devise strategies for different tool materials to operate on batches of different materials. *INTELLIDRILL* decisions could lead to significant savings in tooling costs and reduction in flow line bottlenecks.

DEDICATION

I dedicate this work to my parents Mr. J. Balachandran and Mrs. Bobby Gajalakshmi, who have always been a constant source of inspiration to me. Their perennial efforts in motivating me have kept me going. Without their blessings and encouragement, it would not be possible for me to complete this thesis. I would like to acknowledge and thank my dear sisters Kavitha and Neha for being supportive all the time and helping me in my tough times.

I would like to sincerely express my gratitude to all my professors and lecturers from my undergrad and graduate schools for their invaluable suggestions, guidance and support from time to time. I would like to thank my friends Praveen Herur, MGD, Manish, Santhu, Vishwas, Nagendra, Sickle, Dapuk, Kushal, Modi, Madhuri, Arvind, Scrobbu, TRG, Mani, Ragha, Balu, Nandha, Vivek, Sam, Shankar, Prem, Adi, Sachin, Mithali, VTs, Lakshmi, Anusha, Bikram, Dilip, Anunay, Venu, Sarun, Jumbo, Nitin, NP, Niranjan, Santhosh TH, Hrishi and Sowmya for being such great friends and for their moral support when I needed it the most. I would also like to thank my friends from Oxford English School, Vijaya High School and R.V. College of Engineering.

ACKNOWLEDGEMENTS

The material is based upon the work done for the research paper proposed by the author in association with Dr. Amarnath Banerjee and Mr. Nandhakumar Umashankar, Industrial Engineering Department, Texas A&M University. I would like to acknowledge the constant and continued support of Dr. Amarnath Banerjee and the Industrial Engineering Department. I would like to profusely acknowledge the assistance of my colleague Mr. Nandhakumar Umashankar in helping me develop the material. I would also like to thank Dr. Richard M. Feldman and Dr. Georgia-Ann Klutke for their key inputs to the material.

TABLE OF CONTENTS

CHAPTER Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

Table Page

CHAPTER I

INTRODUCTION

1.1 Overview

This thesis describes the concept of modeling a drilling operation and establishing the different relationships between the machining parameters and material parameters. An attempt has been made in the thesis to model a decision making system in the field of drilling for a Flexible Manufacturing Systems (FMS) layout. Although earlier research has addressed issues like manufacturing productivity, tool failure reduction, tooling cost reduction and total cycle time reduction, they still do not effectively address key issues like adaptive tool selection, extension of tool life and above all modeling a decision making system which can incorporate all these factors. The decision making system has adaptive decision making strategies to reduce bottlenecks in a drilling operation and to extend tool life. The system serves as back end intelligence to drilling machines in FMS layouts for making dynamic and real time decisions. The thesis deals with designing a decision making system for a drilling machine. The decision making system has been implemented using Java with a Graphical User Interface (GUI). The system is modeled such that it can be conveniently and effectively utilized for FMS layouts.

This thesis follows the style and format of the *International Journal of Production Research.*

1.2 Research Preview

Tung *et al.* (1997) present the problems of a Flexible Manufacturing Systems layout at two levels: the Shop Level and the Cell Level. The paper essentially deals with FMS capability of pursuing global benefit of the system under a multi objective, multitask environment. The paper states that FMS flexibility depends not only on flexibility of hardware, e.g. machines and robots but also on the flexibility of the software. This is important because some systems lack the flexibility to be significantly modified after installation (Larin, 1989) and there is a need to allow a human scheduler to influence the final decision. Further, they suggest that FMS scheduling should be able to perform five things:

1. Cooperate with the sequencing mechanism in achieving the system's global goal.
2. Pursue multiple objectives of the system.
3. Provide a quick response with short computation time.
4. Provide flexibility for the human scheduler.
5. Provide a flexible software structure.

A FMS built on a master-slave relationship may have problems of its own. Algorithms at the lower levels of the FMS can only search for solutions under the guidance of algorithms of the higher level of FMS control. The lower level of a FMS is the cell. The cell does not have a prior knowledge of the adjacent cells and there is a lot of interdependency on the master. The local performance of a cell has to be compromised at times to increase the global performance. This uncertain tradeoff is ubiquitous in a master-slave environment.

Duffie *et al.* (1988) state that increased autonomy of the entities of a system reduces the need for a highly intelligent centralized governing body. Autonomy enforces localization of information, isolating each module from other modules in the system. A

nonhierarchical structure also known as a heterarchical structure offers to be a prospect for reduced complexity by localizing information and control. These systems work without a master-slave architecture. The cells cooperate through communication to pursue system goals. Global information is minimized in these systems. This minimization of the global information in the system enhances system modularity, modifiability and extendability. Containment of faults within entities and recovery of faults is other advantage.

The design principles stated by Duffie *et al.* for producing a system of cooperating autonomous entities with a high level of intrinsic modifiability and fault tolerance are:

1. Entities should possess the highest achievable level of local autonomy.
2. Master-slave relationships should not exist between entities.
3. Entities should assume that other entities will not cooperate with them.
4. Entities should delay establishing relationships for as long as possible.
5. Entities should terminate relationships as soon as possible.

In the present system designed and discussed in the thesis, prominence has been given to the decision making at the cell level. In other words, there is a need for a local machine level intelligence instead of total dependency on the master. The system designed should be able to respond in real time and acclimatize to situations without delay. To achieve these goals, there was a need to incorporate the principles of a heterarchical system. Problem addressal for real time and dynamic situations can be enhanced using a heterarchical structure.

1.3 Organization of Thesis

The organization of remainder of the thesis is as follows. Chapter II presents the scope of the problem and the problem description studied in the thesis. Chapter III discusses the modeling environment and the decision variables used to build the *INTELLIDRILL*. Chapter IV deals with the implementation of the *INTELLIDRILL* and the various modules involved in it. Chapter V discusses the results of the different scenarios. Chapter VI presents the conclusions and further scope of the study performed in the thesis.

CHAPTER II

PROBLEM STATEMENT AND SCOPE

2.1 Motivation

Flexible manufacturing system (FMS) problems have been an extensive field of research for the last three decades. FMS is a computer controlled process technology suitable for producing a moderate variety of products in moderate volumes. A FMS is often used in manufacturing components that require several machining operations to be performed in different work stations. A work station can consist of a machine or a robot that performs a particular class of tasks such as drilling holes, bending metal in various directions and so on. Specialized tools are continuously available for one at a time use and changed automatically by computers according to unique requirements for each component as it progresses through the system (Adam and Ebert 1999).

Drilling is one of the several operations performed in a FMS workstation. It is the most common machining and a complex three dimensional cutting operation with the cutting conditions varying along the entire cutting edge from axis to the periphery. The cost of circular hole cutting tools, which exceeds $200 million annually in the United States, represents almost one-third of the cost of all cutting tools (Tool and Manufacturing Engineers Handbook 1976).

Tool selection for a drilling operation has been an active area of research for a long time. The decision to select a particular tool should not only reduce the machining time and the bottlenecks encountered but also extend tool life.

Thus, an integrated system is required that can serve as backend intelligence to drilling machines in a FMS for making dynamic and real time decisions. The following section in this chapter presents the problem considered in this thesis and approach adopted to attain the solution.

2.2 Problem Definition and Scope

Heterarchical system implements control local to the machine or at the cell level. The backend system uses the information present at this level for decision making. The backend enhances the existing level of intelligence present at the cell level. The level of information available at the cell level is sufficient to make real time decisions. With adaptive backend intelligence, real time job scheduling and decision making capability is enhanced. The primary purpose of this thesis is to design the backend for a drilling system which is capable of taking decisions related to tool change, tool selection, speed-feed combination and tool life on its own. The backend uses mathematical and adaptive tool reliability models to simulate the machining conditions and tool availability for an operation. The backend can also be integrated with a Virtual Reality environment for real time visualization of the operation. Such decisions could lead to significant savings in tooling costs and reduction in flow line bottlenecks.

FMS is a complex manufacturing setup and is conceptualized as a constrained interaction of many system variables. The productivity of the system is influenced by the interaction of these variables subject to a set of constraints. One of the primary goals of a FMS is to enhance efficiency and at the same time enable high product quality, subject to the practical constraints. Before designing the backend, the significance of the effect of practical constraints on the efficiency of the manufacturing process should be considered.

In a production system, several bottlenecks in the flow lines are often encountered. Bottlenecks are barriers that affect the smooth flow of material through a channel or a production line. Their magnitude and the problem they present vary from one manufacturing system to another. The productivity of a FMS can be increased by identifying and addressing the bottlenecks that tend to increase the throughput time.

The managerial problem concerns identifying bottlenecks which cause a high degree of imbalance of workload between the machines and ensuring that jobs with a unique requirement for the bottlenecks are served in preference to other jobs which have alternatives available. For this reason, the bottlenecks have been divided into two classes for the system described here: Global and Specific. Global bottlenecks are those processors which currently limit the capacity of the system as a whole. Specific bottlenecks apply more severely to a particular group of jobs and grow more troublesome as the demand for those jobs increases (Upton, 1992).

Global bottlenecks unlike specific bottlenecks fall beyond the scope of local machine intelligence. There are several reasons behind the causes of these bottlenecks:

1) **Factors Associated with Machining:** These are the factors (*Machining Parameters*) attributed to the machining operation. Feed rate, speed and depth of cut some of them. These factors affect the machining time and tend to increase the batch processing time. Feed rate, speed and depth of cut are limited by the material properties of the tool and the raw material. An improper combination of these machining parameters may result in tool failure leading to work stoppage which can cause bottlenecks in the system. The above discussed factors belong to the category of specific bottlenecks and are local to the machine.

2) **Factors Inherent in the System:** These are the factors that are intrinsically present in the material and the machine. Although the magnitude of this factor is

minimal, they tend to introduce inconsistency in the system. Equipment failure, poor material finishes are some of the examples.

3) **Ungovernable External Factors:** These are factors that are beyond the scope of local machine control. In a complex manufacturing setup, allocation of the robots, waiting time for the conveyors etc influence the throughput of the system. Overall, these factors cannot be addressed by the backend.

Bottlenecks increase the Work in Process (WIP) and the system response time, thus leading to a reduction in the overall effectiveness of the system. The intent of this thesis is to deal only on the factors associated with machining for designing the decision making system for a drilling machine. These factors can be controlled to reduce the bottlenecks present in the system with a structured approach to decision making regarding tool selection and machining time allocation. To model a manufacturing environment, it is necessary to understand the system and the key issues the system presents. Some of the key issues are manufacturing productivity, tool wear quantification, tool life estimation, tool failure prediction, tooling costs reduction, total cycle time reduction etc. A decision making system for adaptive tool selection with extension of tool life must address these key issues simultaneously. This is critical because a decision made solely on minimizing the machining time by increasing the feed rate and spindle speed will lead to an increased tool wear or a premature tool failure. A decision to minimize machining time should not increase the tooling costs. This scenario strongly emphasizes the significance of a real time decision making in a manufacturing environment. An incorrect decision could lead to an increased flow time and hence bottlenecks in the system.

The following chapter describes the procedure involved in modeling the system discussed in Section 1.2. It describes the selection of the input and output parameters that form the decision variables for the backend and the modeling environment for the system.

CHAPTER III

MODELING ENVIRONMENT FOR *INTELLIDRILL*

3.1 Overview of the *INTELLIDRILL*

A system is a physical entity bound by a framework built to accomplish a range of goals. It is an organized collection of parts that typically receives inputs, processes it and effectively presents an output/outcome. The system designed in this thesis is called as the *INTELLIDRILL* and incorporates the above principles. *INTELLIDRILL* is a decision making system that determines the feasible machining conditions by selecting the best suited tool based on the given speed-feed combination which is safe for the tool. The system devises strategies for different tool materials to operate on batches of different materials. It is capable of making dynamic and real time decisions by taking into aspect all the conditions pertaining to machining and hence the term *INTELLIDRILL* (*Intelligent Drilling System*).

The basic purpose of the *INTELLIDRILL* is to serve as a backend to a Computer Numerical Control (CNC) / Numerical Control (NC) drilling machine in a FMS layout. *INTELLIDRILL* uses 2 models: a mathematical model to simulate the machining conditions required for a drilling operation and an adaptive tool reliability model to determine the tool availability for an operation.

To design the *INTELLIDRILL* system, the first step is to identify the input and output parameters. These parameters affect the machining operation and are crucial for the smooth functioning of the system. These are the decision variables of the

INTELLIDRILL system. The following section enlists the various input and output variables and discusses these parameters in detail.

3.2 Decision Variables for the *INTELLIDRILL*

A schematic representation of the *INTELLIDRILL* is shown in Figure 1. The Computer Numerical Control (CNC) / Numerical Control (NC) of the drilling machine can be considered as the frontend. The *INTELLIDRILL* constitutes the backend of this drilling system.

The input parameters are the raw material, batch size, the type of the material to be drilled, the hardness of the material to be drilled and the drill geometry. The output parameters are the machining parameters. The machining parameters that generally affect a drilling process are:

- Feed rate
- Depth of cut
- Cutting speed
- Material removal rate (MRR)
- Cutting fluids

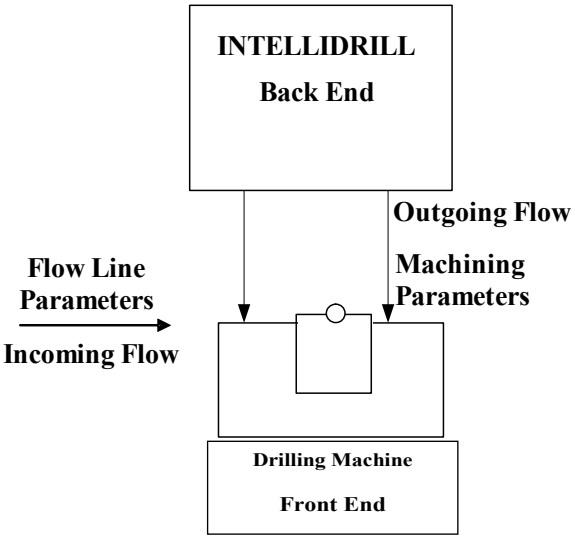

Figure 1. Schematic Representation of *INTELLIDRILL*

The first four variables are considered for designing the *INTELLIDRILL*. Cutting Fluids help in the removal of chips, lubrication and in turn minimizing the tool wear rate. Varying the quantity of the cutting fluid would not help in reducing the cycle time. Mathematical modeling of the effect of cutting fluids on machining is complicated and is neglected. Hence, cutting fluid has not been considered in designing the system.

Feed rate of a drill is the distance the drill moves into the work at each revolution of the spindle. It is expressed as feed per minute. The feed per minute may be defined as the axial distance moved by the drill into the work per minute (Sandvik Coromant 1996).

$$\text{Feed rate} = f \times n \quad \frac{mm}{\text{sec}} \qquad (1)$$

$$s = \frac{f}{Z} \frac{mm}{rev} \qquad (2)$$

where

f = feed

s = feed rate

n = Spindle speed

Z = Number of cutting edges

Cutting Speed refers to the peripheral speed of a point on the surface of the drill in contact with the work. It is usually expressed as distance/unit time.

$$\text{Cutting Speed} = \pi \times d \times n \quad \frac{mm}{\text{sec}} \qquad (3) \qquad \text{where}$$

Depth of cut in drilling is equal to one half of the drill diameter. In drilling, feed rate is usually the combination of feed and depth of cut. The machining process is defined by Material Removal Rate (MRR), feed rate and cutting speed. MRR is defined as volume of the material removed per unit time. MRR is directly proportional to feed rate and cutting speed.

$$\text{Material Removal Rate} = \frac{V}{t_{drill}} \qquad (4)$$

$$= A \times (\pi\, d\, n) \quad \frac{mm^3}{\text{sec}} \qquad (5)$$

$$\text{Cross sectional area } A = \left(\frac{d}{2}\right) \times s \quad \frac{mm^2}{rev} \qquad (6) \qquad \text{where}$$

V = Volume of the hole to be drilled

The volume of material V removed can also be calculated from drill geometry.

$$\text{V} = K\left[\frac{\pi\, d^2}{4} \times (h)\right] \quad \frac{mm^3}{\text{sec}} \qquad (7) \qquad \text{where}$$

K = Correction factor

h = Depth of drill

Drilling is a hole making operation. It does not produce a true hole and hence a correction factor is used. The hole diameter is not uniform throughout the depth of the

hole i.e., the hole may be bell mouthed, ball-shaped, concave or the hole axis may be inclined. Without special precautions, the drilling process will yield a maximum of 0.004 in. in error of slope at a penetration of 1 ½ in. (Tool and Manufacturing Engineers Handbook 1976). The error of the slope is negligible for calculation of the MRR. Hence, the value of K has been assumed to be 1.0 to consider the hole drilled as a completely cylindrical hole.

The input variables have to be translated into a rationale set of output variables. This is the function of the backend. The backend accepts the input parameters also known as the flow line parameters, computes the speed-feed combination that is safe for the tool, the Material Removal Rate (MRR), the best suited tool for the particular batch size, the availability of the tool (new tool or old tool) and also calculates the adequate number of tools required. A check is made to ascertain whether the forces affecting the drilling operation are under permissible limits.

3.3 Modeling the Drilling Operation

INTELLIDRILL must have an insight on the drilling process to effect proper decisions. Thus a mathematical model of the drilling operation needs to be developed. An operation is defined as a value addition process at a machining station. Operation is measured by cycle time. *Cycle Time* is defined as the time a part enters the system to the time it exits the system. Cycle time comprises of setup time (time to setup the work piece for machining and the waiting time) and the actual machining time. Setup time has a variable and a fixed component. The fixed component is due to limitations in system like table traversing speeds, setting up the new tool, replacing the old tool etc. The variable component includes delay in waiting for the next part/batch to arrive. This component is usually a small negligible quantity and has not been considered for calculation purposes.

$$T_{op} = \frac{B_N}{P_R} \times \frac{1}{N_H} = t_{setup} + t_{drill} \qquad (8)$$

$$t_{setup} = t_{const} + t_{\text{var}} \qquad (9)$$

$$t_{drill} = \left(\frac{B_N}{P_R} \times \frac{1}{N_H} \right) - t_{const} - t_{\text{var}} \qquad (10)$$

where

B_N = batch size

N_H = Number of holes to be drilled

P_R = Production rate

t_{var} = Variable cycle time component

t_{const} = Constant cycle time component

From (4), the Volume of the Hole to be drilled is also given by

$$V = MRR \text{ x } t_{drill} \qquad (11)$$

From this, $$t_{drill} = \frac{K\,h\,z}{2\,f\,n} \text{ sec} \qquad (12)$$

Machining is defined as removal of required material by application of an effective cutting force. This cutting action leads to development of forces. The forces generated are Axial / Thrust force, Radial force and Shearing force. The Shearing force is induced by the torque. The stresses generated in material have to be within the limits of the tool's elasticity properties for machining feasibility. In drilling the Radial forces cancel out due to symmetric nature of the cutting operation. The mathematical model for the Thrust force and the torque generated in a drilling operation is given by Shaw and Oxford Equations (HMT 1997).

Thrust force (P_X) $$\frac{P_X}{d^2 H_B} = 0.55 \frac{s^{0.8}}{d^{1.2}} \left[\frac{1-k}{(1+k)^{0.2}} + 2.2k^{0.8} \right] + 0.07k^2 \qquad (13)$$

Torque (M) $$\frac{M}{d^3 H_B} = 0.03 \frac{s^{o.8}}{d^{1.2}} \left[\frac{1-k^2}{(1+k)^{0.2}} + 3.2k^{1.8} \right] \quad (14)$$ where

M = Torque ($Kgfcm$)

P_X = Axial Thrust force (Kgf)

d = Diameter of the drill (mm)

H_B = Hardness of the material

k = Chisel edge length/drill diameter

The above equations indicate that the thrust force and Torque are dependent on the hardness of the material and drill geometry.

$$P_X \leq \frac{\sigma_{X\max} \times Area}{FOS} \quad (15)$$

Torque $M(Kgfcm) = \frac{P_Z}{2} \times \frac{d}{10}$

$$M = \frac{P_Z \times d}{20} \quad (16)$$

$$P_Z \leq \frac{\tau_{XY\max} \times Area}{FOS} \quad (17)$$ where

FOS = Factor of safety

P_Z = Shear force (Kgf)

$\sigma_{X\max}$ = Ultimate crushing stress

$\tau_{XY\max}$ = Ultimate shear stress

$Area = \frac{\pi d_{eff}^{\ 2}}{4}$ where

d_{eff} = Effective diameter

3.4 Modeling the Tool Life

In any machining operation the tool undergoes wear because of the cutting forces involved and the nature of the tool material. The tool is useful until the tool wear goes beyond a practically determined permissible value. This duration is termed as tool life. A standard tool life is specified by the manufacturer for machining a specific material. These are however subjected to standard machining conditions. Prediction of remaining tool life is critical to the decision making process. Remaining tool life ensures that the tool is available for an operation and proper tool allocation is possible.

An improper feed-speed selection could result in tool failures. A decision with such an improper combination could cause premature tool failures during the operation. In order to avoid this, the model predicting the tool failure needs to be adaptive. An adaptive model ensures that improper decisions presently taken can be used for better decision making in the future.

Tool failures are generally modeled using the Weibull Distribution. It is the most widely used distribution in reliability engineering for modeling failures caused by fatigue, corrosion, mechanical abrasion, diffusion, and other degradation processes. For *INTELLIDRILL,* Weibull distribution is not suitable. In the Weibull failure model, the model becomes dependent on the distribution i.e. tool life becomes dependent on the weibull parameters: shape and scale. This model is not adaptive. This problem is overcome by using the empirical distribution. The advantage with the empirical distribution is that it resembles of the Cumulative Distribution Function (CDF) of the original distribution once it has sufficient data points. A Cumulative Distribution Function gives the probability that a random variable X is less that a given value x i.e. F (x) = Pr{X<=x}. If ‘n’ data points constitute the empirical distribution, the cumulative probability at each step is increased by 1/n. The cumulative probability increases from 0 to 1 in the ‘n’ step intervals specified.

The tool life model for *INTELLIDRILL* developed using an empirical distribution is predicted with initial set of data points. If the tool undergoes a premature failure based on the decision, it is included as a data point. If a series of decision made causes the tool to fail, then the empirical distribution adapts to these failures by reducing the mean life of the tool. The decision making capability improves with every decision that has been taken by the *INTELLIDRILL*.

$$T \sim F(t) \qquad (18)$$ where

T - Time for failure of the Tool

F (t): Empirical Distribution $F(t) = P\{T \le t\}$

$$P\{T > t \mid T > t_o\} = \frac{1 - F(t)}{1 - F(t_o)} \qquad (19)$$ where

t_o = time for which the tool has already been used.

$t = t_o$ + Machining Time Allocated.

The output of *INTELLIDRILL* is the particular type of tool that has to be used for drilling a batch of material and the number of such tools required. The system is capable of handling multiple materials and using multiple tools.

3.5 Factor of Safety and Critical Failure Probability Limit

The stress in a material should always be less than the permissible stress. Factor of Safety (FOS) is a factor or an allowance used to cushion any process/product to withstand failure stress or stresses that exist. FOS provides more endurance and stability to the process/product.

Permissible Stress = (Maximum Ultimate Shearing/Crushing Stress)/ FOS (20)

High FOS ensures low risk and reduction in permissible stress. Correspondingly, there may be an increase in cost and sometimes the operation may be rendered machining infeasible. With a low FOS, there is always a high risk and the impending danger of the operation getting rendered as infeasible. A check is made to ensure the process always stays above a fixed level of FOS. If the operation is carried out above the Threshold FOS, then the operation is rendered feasible. Below the threshold FOS, the incoming material cannot be drilled.

Critical Failure Probability Limit also known as the critical limit is a tool function to compute the availability of the tool. If the availability of the tool is not within permissible limits, the tool is deemed unavailable for the operation. The permissible limit is generated by simple inverse empirical function which returns the permissible time at the maximum acceptable probability of failure. The critical failure probability limit is set by the user/management.

3.6 Decision Modules of the *INTELLIDRILL*

The objective of the *INTELLIDRILL* is to select a set of tools to perform a drilling operation on a given batch ensuring that there is a reduction in the premature tool failures and tooling costs.

At every point in the decision making process, it must be ensured that the operation is machining feasible. This is one of the sub objectives of the *INTELLIDRILL*. An operation is machining feasible if the stresses generated are within the permissible limits. An iterative procedure to determine a feasible FOS is followed. This procedure ensures low risk in the operation. It initially starts with a very high FOS.The system should check if the operation factor of safety is greater than the threshold. If the stresses generated are not within permissible limits, the factor of safety is stepped down. The procedure is repeated until the operation is machining feasible. If the operation factor of safety dips below the threshold, then the operation cannot be completed with the given type of tools. The system should ensure that the machining is feasible under the highest factor of safety for the given input.

The second sub objective of the *INTELLIDRILL* is to estimate the tooling requirement for a given operation with reduced tooling costs. In a multiple tool type scenario, the best suited tool for an operation has to be selected. In order to reduce the tooling costs, a proper combination of new and old tools should be used.

The third sub objective of the *INTELLIDRILL* is to estimate the machining parameters and check whether they are under permissible limits. *INTELLIDRILL* has to ensure that the given batch has to be machined within the specified processing time. It computes the feed-speed combination to meet this sub objective.

The final sub objective is to check for the availability of the tool(s) for an operation. If atleast one tool is not available for the operation, the existing tooling requirement has to be changed to accommodate the availability of the tools.

In order to achieve the overall objective, the decision making process has to accomplish the sub objectives. Decision making modules are created to handle each sub objective. Once the sub objective of a module is met, the decision making is transferred to the next module. This procedure is repeated cyclically between the modules until all the sub objectives are accomplished. The modules work in a serial order. The modules are designed in such a way that the goal of one module is not dependent on the goal of the other. Each module ensures a local optimum is achieved. Once the local optimum has been achieved, the modules have accomplished their tasks and an overall decision is made. The modules do not work in the tandem and thus it is difficult to obtain a global optimum solution.

Module 1 checks for the machining feasibility of an operation. This module uses the Mathematical Force Model for estimating the stresses generated during the operation. It ensures a feasible FOS for a given operation.

Module 2 estimates the number of tools required for an operation. Based on the tool usage and the tool availability, either a new or an old tool is chosen for the operation. This module also ensures a proper combination of new and old tools to reduce the tooling costs.

Module 3 estimates speed, feed and MRR for an operation provided they are within permissible limits. The module determines a feed-speed combination that helps in estimating a feasible MRR.

Module 4 checks the availability of a tool for an operation. This module uses the Reliability Model of tool failures. The module is initialized with 50 data points for fair and accurate decision making. The Reliability Model uses an empirical distribution (discussed in Section 3.4) to predict tool life. The inverse empirical function of the Reliability Model yields a permissible time at the maximum acceptable probability of failure. The module checks for the tool availability if it falls within/above the permissible limit. If the tool undergoes a premature failure based on the decision, that failure is recorded as a data point. The module then recomputes the mean life and the new mean life for the tool is updated for the next decision. Thus, this module enables the adaptive mechanism of the *INTELLIDRILL*.

The following chapter describes the implementation of these decision modules and a case analysis focusing the adaptive nature of the *INTELLIDRILL*.

CHAPTER IV

IMPLEMENTATION FRAMEWORK

4.1 Overview of the Implementation Strategy

The *INTELLIDRILL* system has been programmed in Java. The information and data needed for the system is stored in a database (Microsoft Access). The *INTELLIDRILL* has been implemented using a 5-Level decision structure as shown in Figure 2. At the lowest level (Level-0), it comprises of the basic two classes: Tool and Batch. The 2nd layer (Level-1) is the Operation class which implements all drilling functions. The 3rd layer (Level-2) comprises of various decision modules. The 4th layer (Level-3) is the *INTELLIDRILL* decision module. The final level (Level-4) is the Graphical User Interface (GUI). The following sections (4.2 - 4.6) discusses in detail the composition and function of each level.

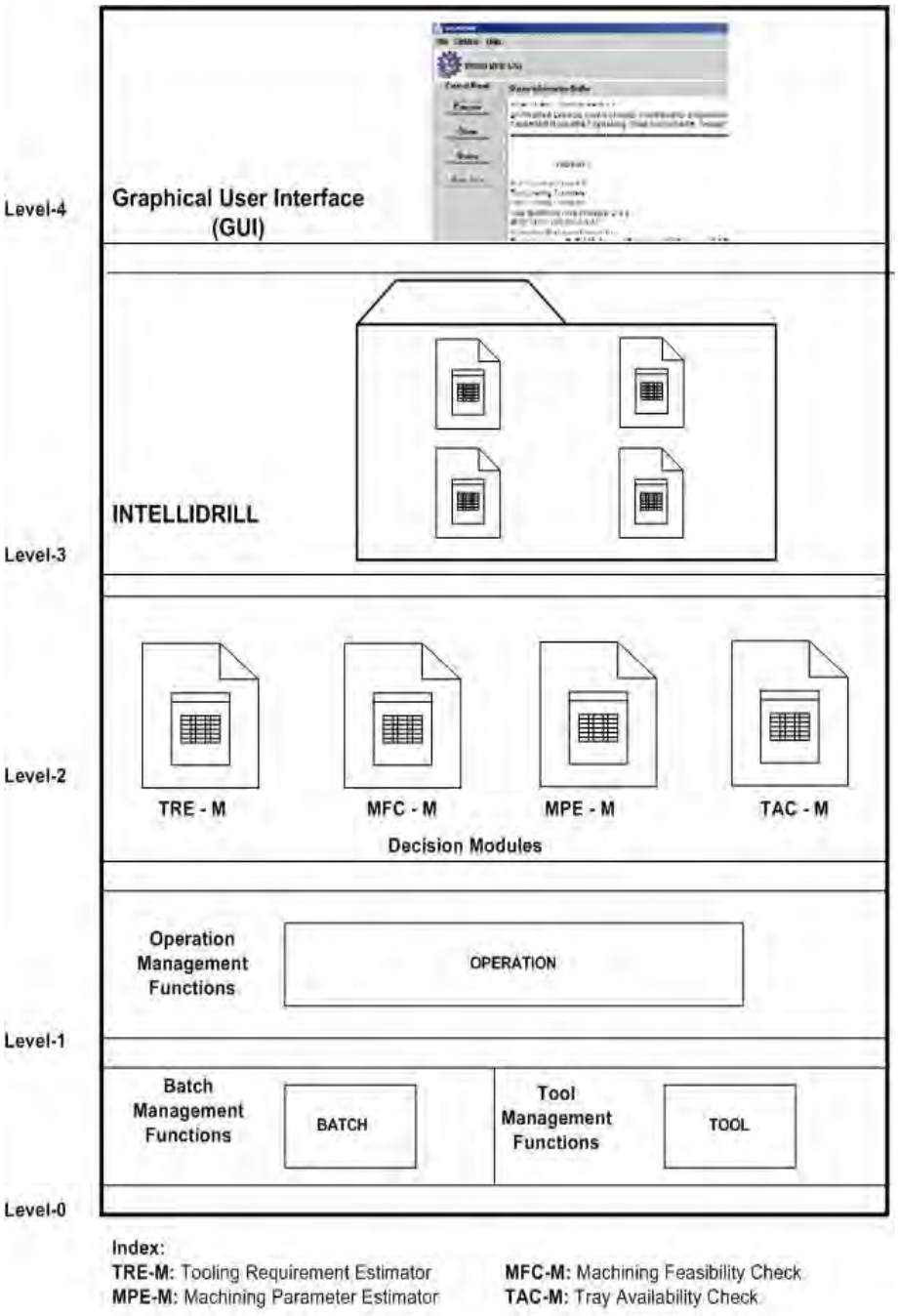

Figure 2. Five Tier Decision Structure

4.2 Level – Zero: The Tool and the Batch Class

Level-0 implements the Tool and Batch Class.

4.2.1 Tool Class

The Tool Class implements all the Tool Management functions. Tool Class contains information about the tool currently being used by the operation. This class connects to the tool database to load /unload tool for decision making. This class also checks if the instance is available for an operation.

An instance of the Tool Class is defined by the following member variables:

TOOLID: Tool Identification Number.
ToolStatus: Tool Status [NEW/OLD/USE].
ToolUsage: Tool Usage Time (min).
ToolDiameter: Tool Diameter (mm).
ToolType: Tool Material Type [HSS/Carbide].
ToolChangeOverTime: Tool Change Over Time (min).
ToolUltimateCrushingStress: Tool Material Ultimate Crushing Stress. (N/mm^2)
ToolUltimateShearingStress: Tool Material Ultimate Shearing Stress. (N/mm^2)
ToolMeanLife: Tool Mean Life (min).
ToolChiselEdgeLength: Tool Chisel Edge Length (mm).
ToolMachiningTime: Assigned Tool Machining Time (min).

More information is available in Tool Class API documentation in Appendix A.

4.2.2 Batch Class

The Batch Class implements all the Batch Management functions. Batch Class contains information about the batch currently being used by the operation. This class connects to the schedule database to load /unload batch for decision making. It also estimates the machining time to be allocated to the batch and to prepare the batch for an operation.

An instance of the Batch Class is defined by the following member variables:

BATCHID: Batch Identification Number.
BatchSize: Size of the Batch.
BatchMaterialHardness: Batch Material Hardness (Indicated by the Brinell hardness number {BHN}).
BatchTime: Time allocated to process the entire batch (min).
BatchStatus: Status of the batch [NEW/OLD/USE (indicates decision is currently taken on the batch)].
BatchUnitSetupTime: Unit Setup Time (min).
BatchUnitMachiningTime: Unit Machining Time (min).
BatchTotalMachiningTime: Total Machining Time for the batch.
BatchHoleHeight: Hole Depth.

More information is available in Batch Class API documentation in Appendix B.

4.3 Level – One: The Operation Class

Level-1 implements the Operation Class. Level-1 is interaction level for Level-0 components.

4.3.1 Operation Class

Operation Class is an interaction level for Batch and Tool Class. Operation implements an instance of Batch as _Batch which indicates the current batch being processed by the Operation and a stack of Tool as Tray. _Tool is a temporary tool used for initialization which is also an instance of the Tool Class.Operation class implements the Mathematical Force Model which are included by the function getThrustForce() and getTorque(). The machining feasibility check is based on these methods. This class allows multiple instances of operations with different tool types, as clones for comparison.

An instance of the Operation Class is defined by the following member variables:

OPERATIONID: Operation Identification Number.
_Tool: Temporary Tool.
Tray: Stack of Tools for an Operation.
_Batch: Batch processed in the current Operation.
Speed: Operation Spindle Speed.
Feed: Operation Feed Rate.
DepthOfCut: Operation Depth of cut. .
MaterialRemovalRate: Operation Material Removal Rate.
Status: Status of Operation.
FactorOfSafety: Operation Factor of Safety.

THRESHOLD: Threshold Factor of Safety [Minimum].

stepValue: Step value to reduce the Factor of safety during analysis.

ThrustForce: Axial Force calculated from the Mathematical Drilling Model.

ShearForce: Shear Force calculated from the Mathematical Drilling Model.

StepUpSpeed: Step value to increase the spindle speed.

StepUpFeed: Step value to increase the feed rate.

SpindleSpeedLimit: Maximum permissible spindle speed.

FeedRateLimit: Maximum permissible feed rate.

_Tolerance: Approximation Tolerance.

More information is available in Operation Class API documentation in Appendix C.

4.4 Level – Two: The Four Decision Modules

Level-2 comprises of four decision modules. These decision modules access methods defined in Operation, Tool and Batch Class. These modules are independent of each other at this level. Each module accomplishes the previously stated sub objectives. The modular components of this level are:

MFC-M: Machining Feasibility Check Module.

TRE-M: Tool Requirement Estimator Module.

MPE-M: Machining Parameters Estimation Module.

TAC-M: Tool Availability Check Module.

4.4.1 MFC-M: Machining Feasibility Check Module

This module determines if the current operation is feasible for machining. The decision variables are based on the tool type, batch material hardness, drill geometry and factor of safety for an operation. The module ensures that the machining is feasible under the highest factor of safety for the given input. The logic flow for this module is shown in Figure 3.

4.4.2 TRE-M: Tool Requirement Estimator Module

This module estimates the tooling requirement for a given operation. It accepts the machining time for which the tooling requirement has to be estimated. TRE-M interacts with level-0 Tool methods to load and unload instance of Tool. Total machining time is broken down into time slots which can be allocated to a new tool. This is based on the mean life of the tool which is inferred from the empirical distribution. If the allocation time is less than the mean life of the tool, then an old tool whose available time is just greater than the allocation time is loaded into the tray. TRE-M determines the machining setup for the operation.

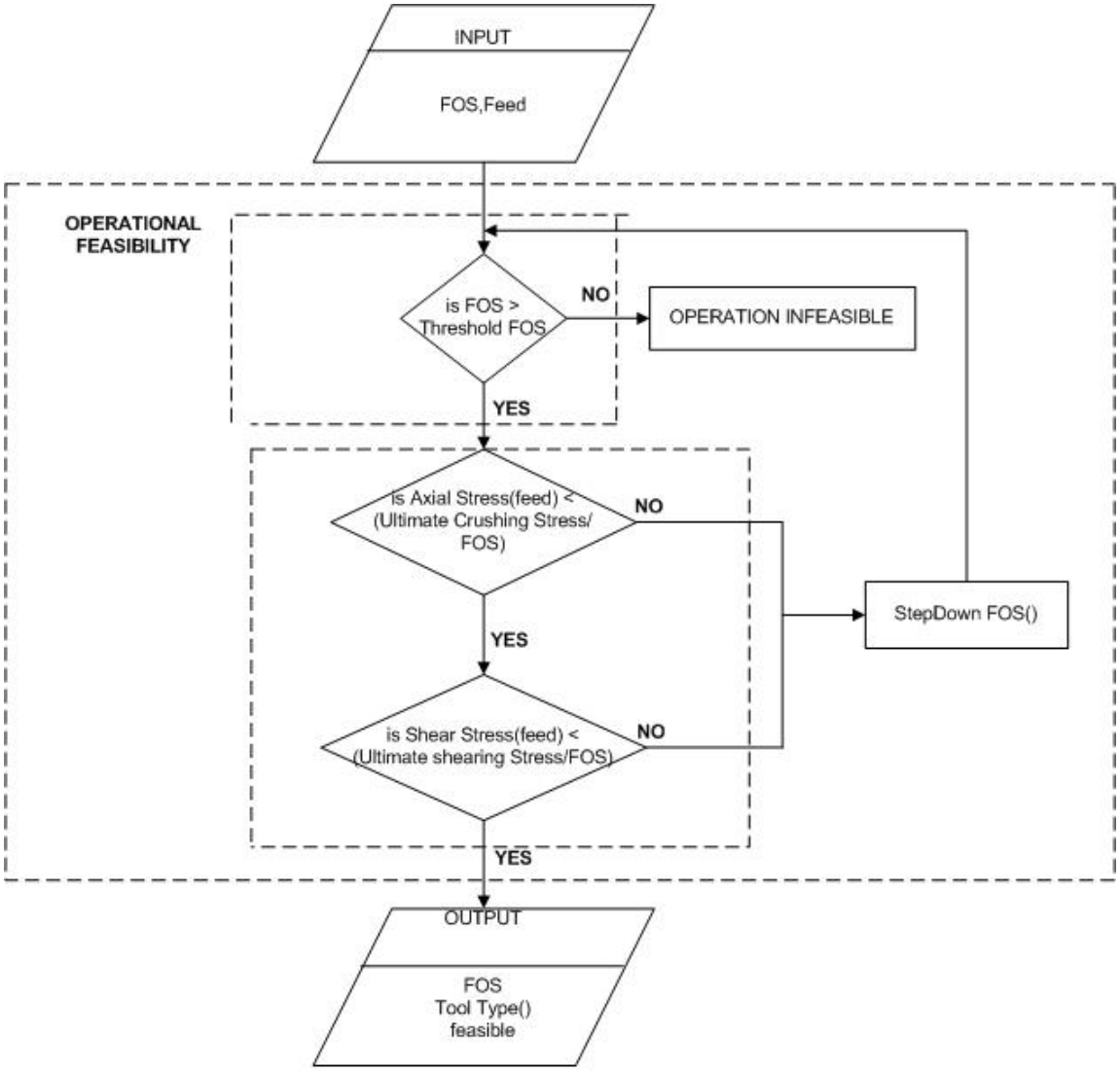

Figure 3. Machining Feasibility Check Module (MFC-M)

TRE-M performs a cost analysis on all the tools that are feasible for the operation based on the method **ToolCostAnalysis (n1, n2, n3…nk).** The function takes into account the unit cost of tool, the cost weight factor and factor of safety for an operation and computes a simple ratio to effect a decision. TRE-M finalizes the process by adjusting the Total Machining Time for the operation taking into account the tool change over time during the operation. The output of this module is adjusted Total Machining Time and Number of Tools for each Operation. The logic flow for this module is shown in Figure 4.

4.4.3 MPE-M: Machining Parameters Estimation Module

This module estimates the machining parameters for an operation. MPE-M accepts Total Machining Time as input and computes the Unit Machining Time (UMT). The Material Removal Rate (MRR) is estimated based on UMT and the hole geometry. The module checks if this MRR is feasible. If feasible, it computes the feed rate and spindle speed combination that offers the closest approximate to the required MRR. The module returns the computed feed and speed as output. The logic flow for this module is shown in Figure 5.

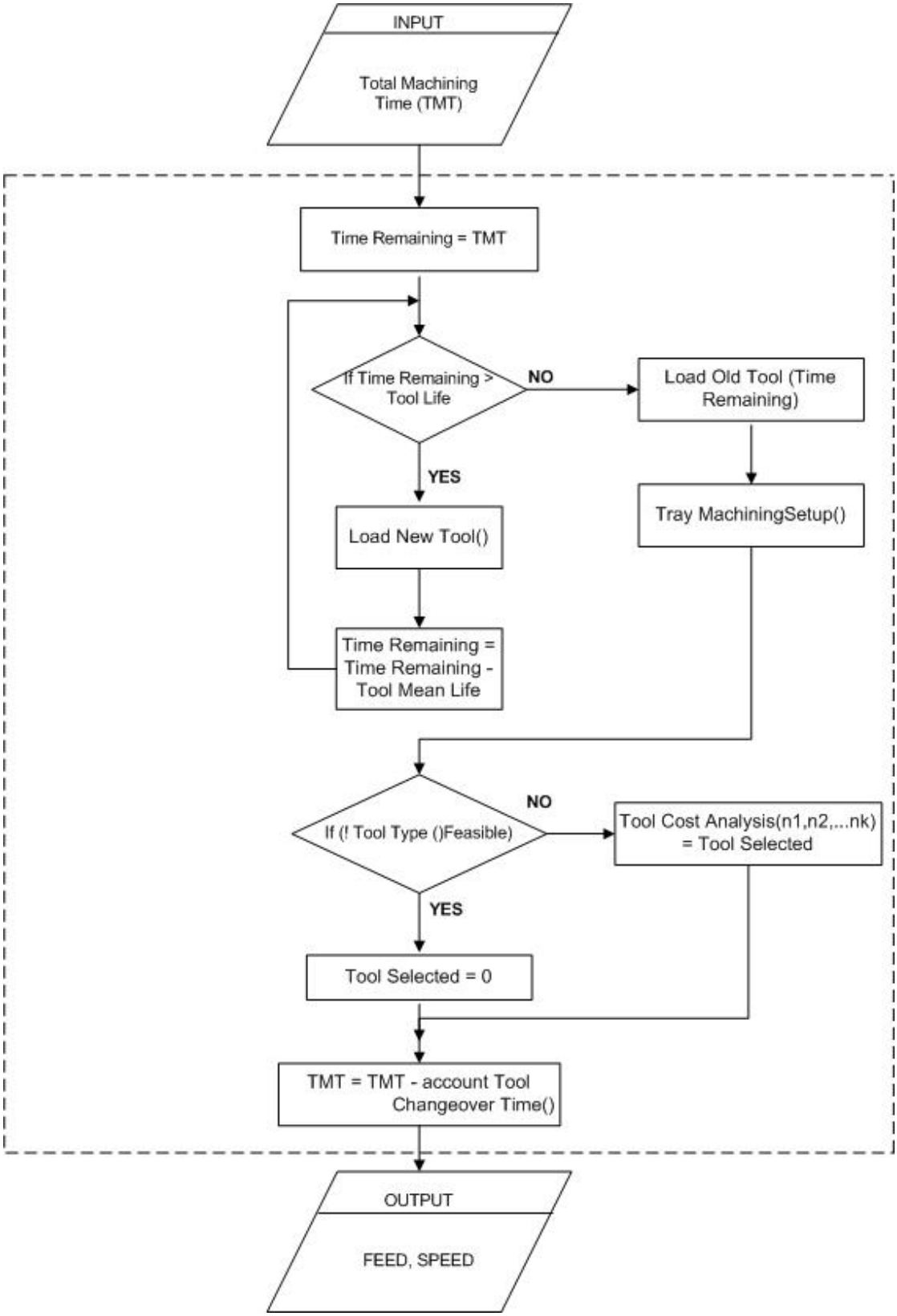

Figure 4. Tool Requirement Estimator Module (TRE-M)

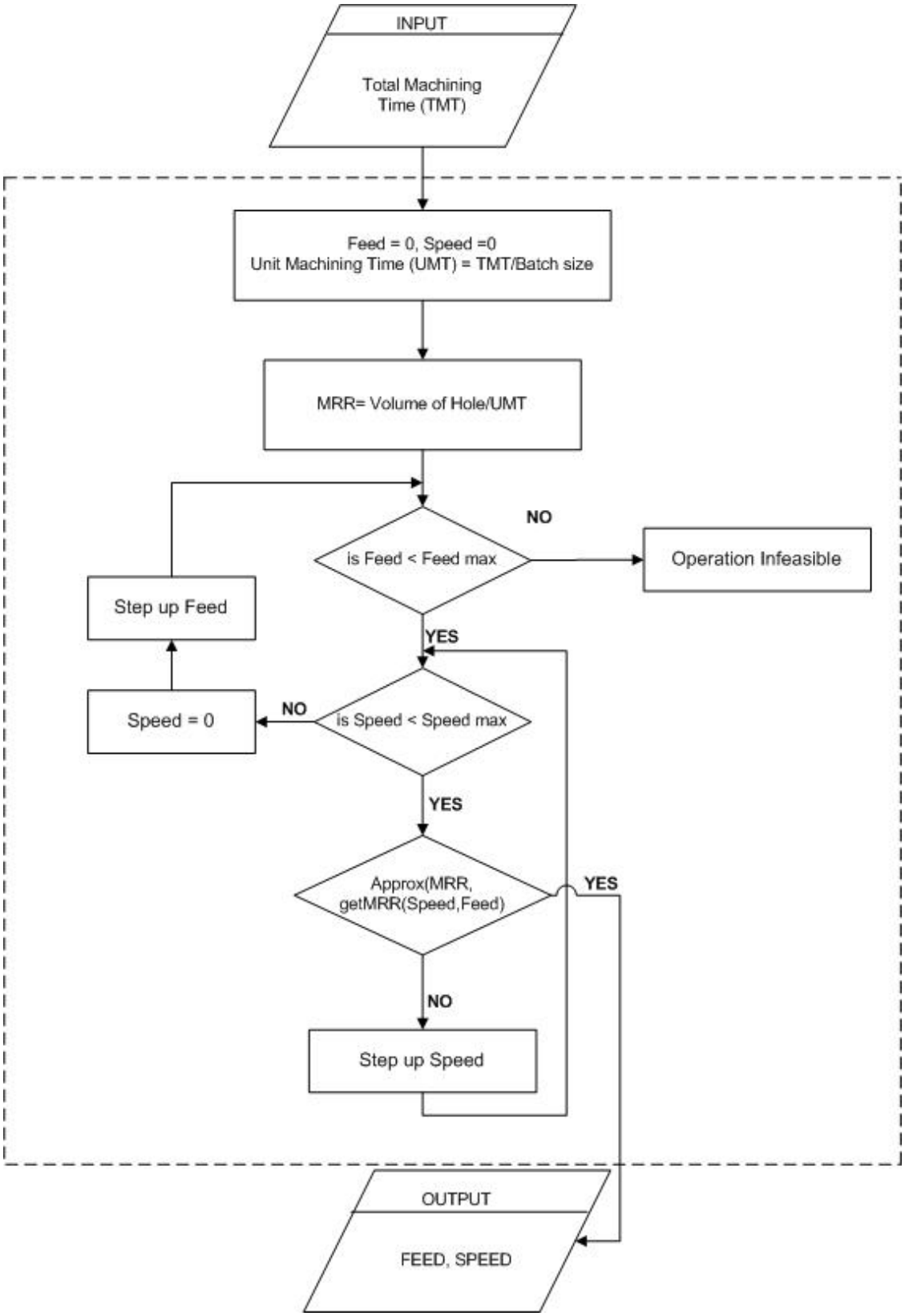

Figure 5. Machining Parameters Estimation Module (MPE-M)

4.4.4 TAC-M: Tool Availability Check Module

This module checks the availability of the Tray for the operation. TAC-M implements the Reliability Model of the tool failures. It interacts with level-0 Tool function to compute the availability of the tool for operation. If the availability of the tool is not within permissible limits (Discussed in section 3.5), the tool is deemed unavailable for operation. This implies that the machining time allocated to the tool extends beyond permissible time. Hence the tool machining time is adjusted to maximum permissible time and the total slack for the tray is calculated. The tray is scanned for old tools to accommodate the total slack. The slack time is adjusted with the old tools. TAC-M returns the remaining slack time that needs to be allocated. The logic flow for this module is shown in Figure 6.

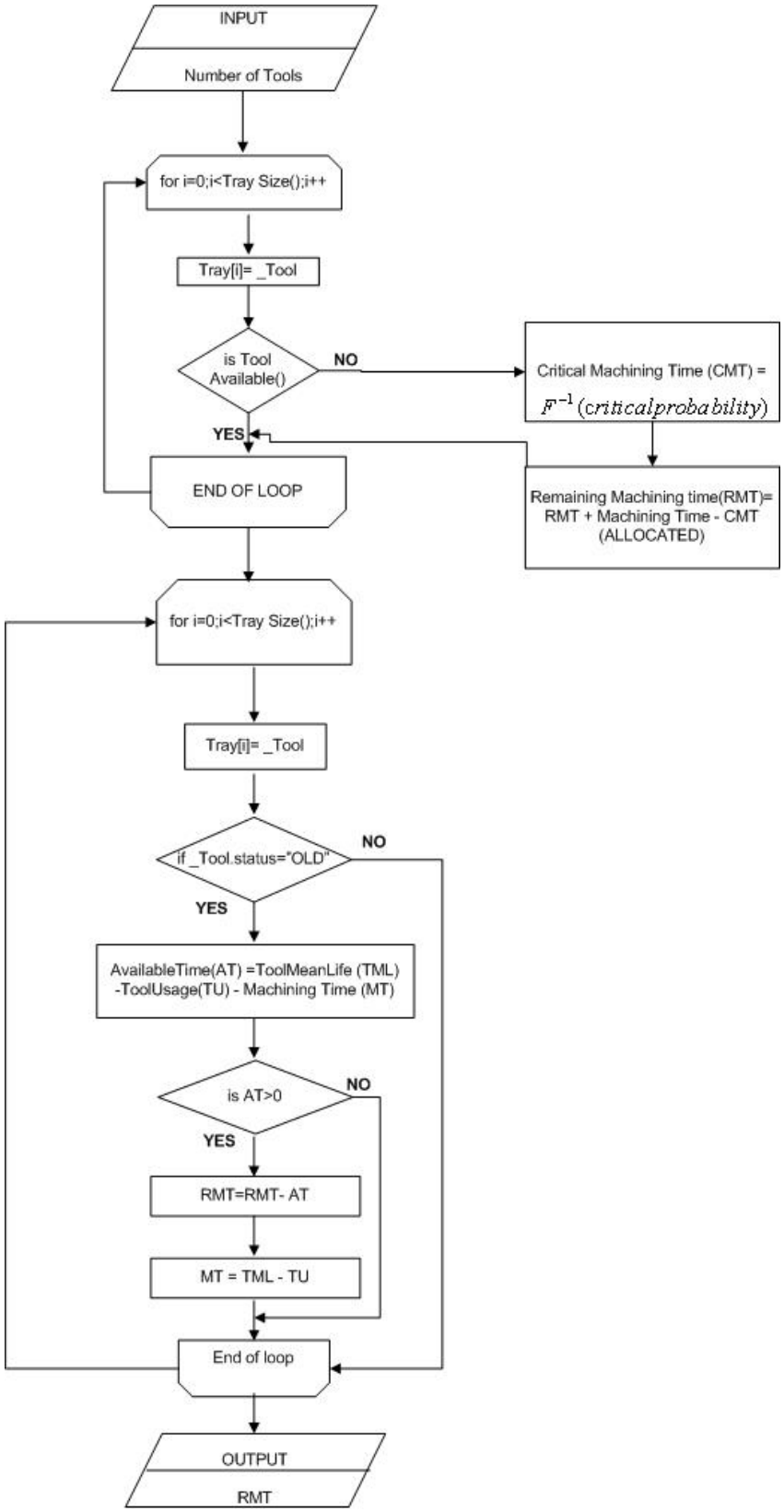

Figure 6. Tool Availability Check Module (TAC-M)

4.5 Level – Three: *INTELLIDRILL*

This level defines the *INTELLIDRILL* decision module. This level permits interaction of Level-2 components.

4.5.1 *INTELLIDRILL* Module:

A special case discussed here has only two types of drill bit materials: High Speed Steel (HSS) and Carbide. These two tools will be enough to address all the problems that will be encountered by the *INTELLIDRILL*. These two tools encompass the whole spectrum of tools. The HSS drill bit constitutes the lower end of the spectrum. Almost all materials can be drilled by HSS drill bit. Those materials that do not fall in this category will be drilled using the carbide drill bit.

This module forms the interaction layer of the four level-2 modules. The system is initialized with _OperationHSS and _OperationCARB. _OperationHSS simulates the operation using HSS Tools and _OperationCARB with Carbide Tools. The initial operation parameters are estimated if the batch processing time is provided. If no batch processing time is provided, the module iteratively computes the minimum processing time to complete the batch. The minimum processing time is computed by starting with an initial batch processing time which is not machining feasible and increasing it until it reaches machining feasibility.

The module then performs feasibility check using MFC-M. Operation is declared infeasible if it is neither HSSFeasible nor CARBFeasible. If the operation is feasible, the module uses TRE-M to estimate the initial tooling requirement. A cost wise tooling decision is taken at this stage.

The module then estimates the machining parameters based on the total machining time adjusted with the tool change over time using MPE-M. The feasibility of the current operation is checked using MFC-M. If the operation is feasible, then TAC-M checks the tray availability and computes the remaining machining time. If the remaining machining time exists then the module recursively iterates between TRE-M, MPE-M, MFC-M and TAC-M until the remaining machining time is 0.

The decision is taken based on feasibility of _OperationHSS and _OperationCARB. If operation is both HSSFeasible and CARBFeasible, the tool type selection is based on the tool cost analysis performed by the TRE-M. The required operation is selected based on feasibility and the final values computed are returned as the final decision. The module returns Operation type [HSS/Carbide], feed, speed and the Tool Tray Machining setup. The logic flow is depicted in Figure 7.

MFC-M computes the maximum possible factor of safety for a given operation. TRE-M estimates the minimum tooling requirement for the operation. MPE-M and MFC-M estimate the maximum possible feed-speed combination. TAC-M ensures minimum probability of failure. *INTELLIDRILL* uses this feature to iteratively determine the feed-speed combination that satisfies the required batch processing time with minimum possible tooling requirement and maximum possible safety.

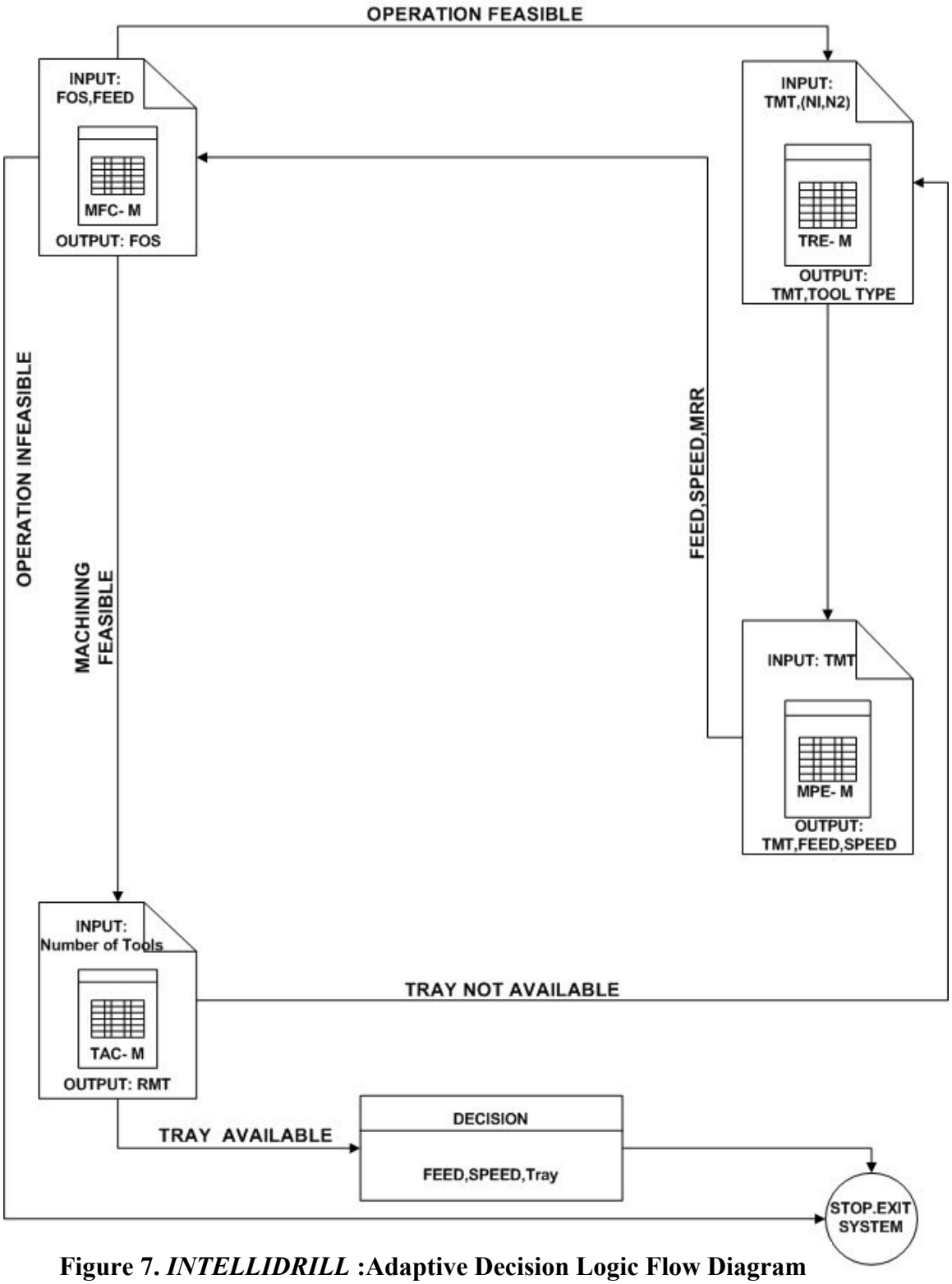

Figure 7. ***INTELLIDRILL*** **:Adaptive Decision Logic Flow Diagram**

4.6 Level – Four: The GUI (Graphical User Interface)

This feature has been added to provide flexibility to the user in real time decision making. GUI allows the user to set the decision parameters. The GUI consists of a Status Buffer that displays the flow of the decision-making process. The program also allows the entire decision making process to be logged to a file specified by the user. Various options can be set using the options dialog. This dialog allows the user to set the options regarding operation, tools, Cost analysis, spindle speed, feed, FOS, database etc.

The GUI is built using a java application called Netbeans IDE. The NetBeans platform is a powerful modular framework used for building powerful modular applications. The platform project provides a language neutral framework, which can be used as a basis for creating any rich desktop application and IDE (Integrated Development Environment) development. On top of the NetBeans platform is an "IDE Platform" which introduces IDE specific functionality: Projects, generic editor or debugger support. Figures 8-12 show some of the screen shots of the GUI.

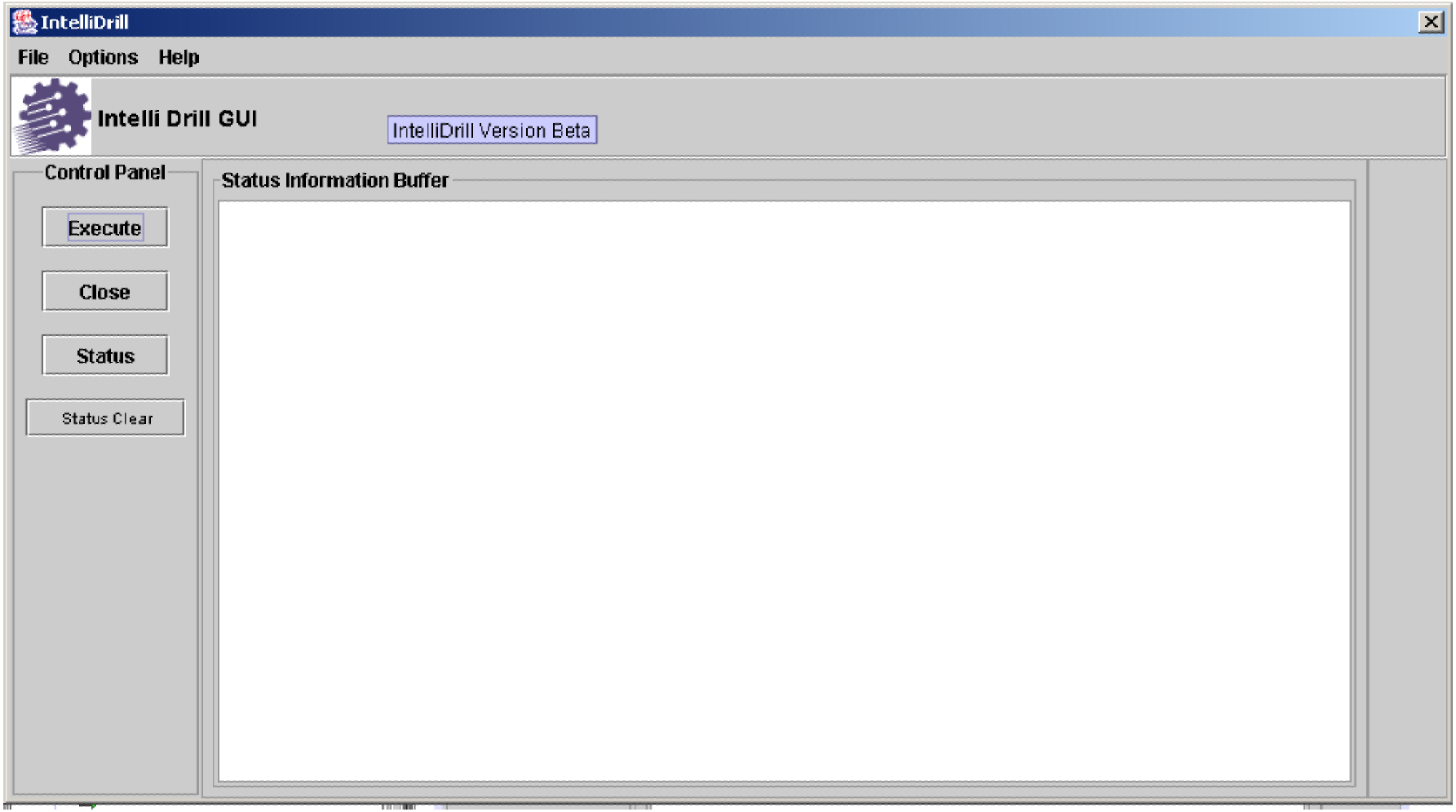

Figure 8. ***INTELLIDRILL*** **: Graphical User Interface (GUI)**

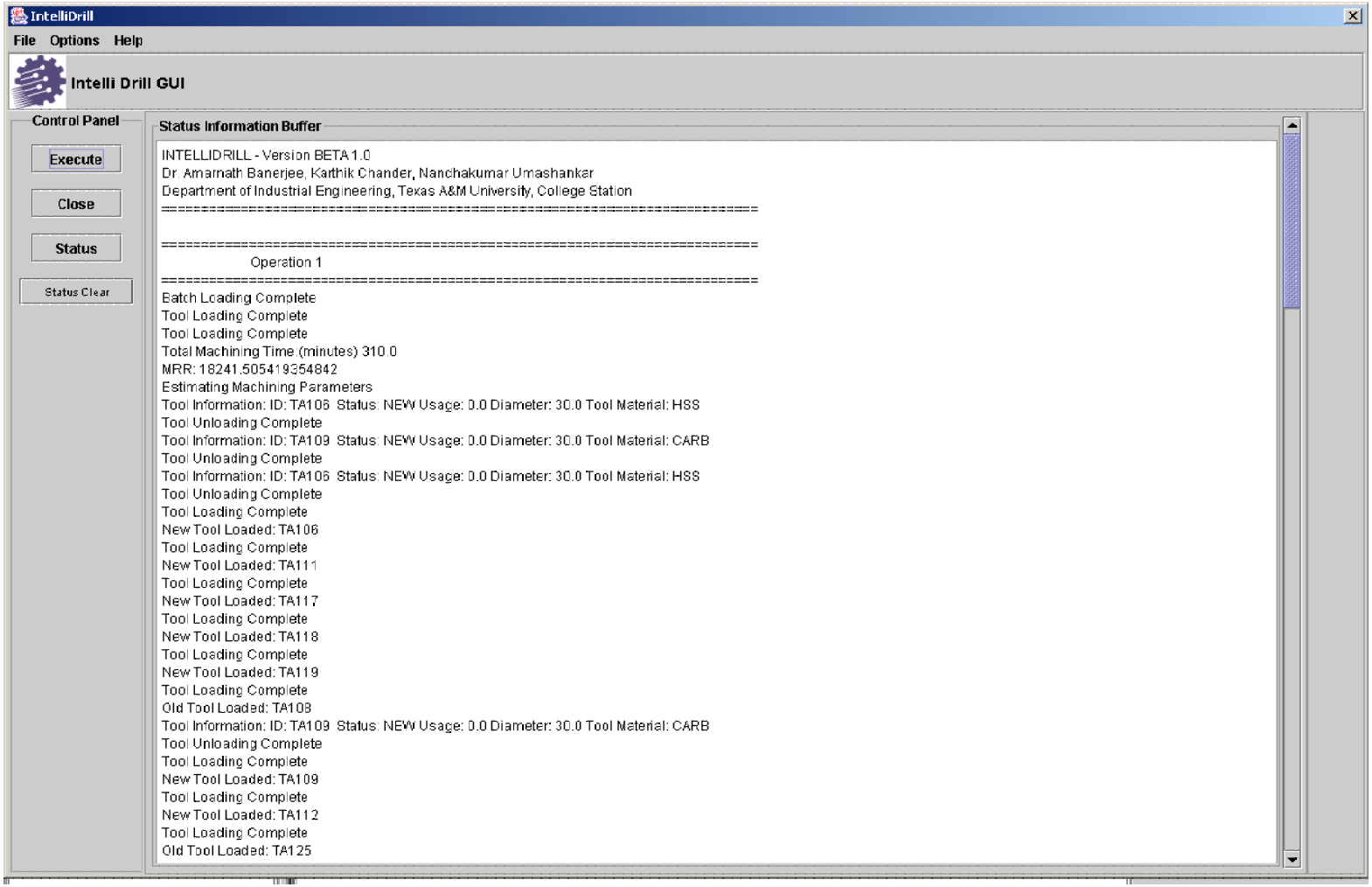

Figure 9. *INTELLIDRILL*: GUI Displaying Status Buffer Information

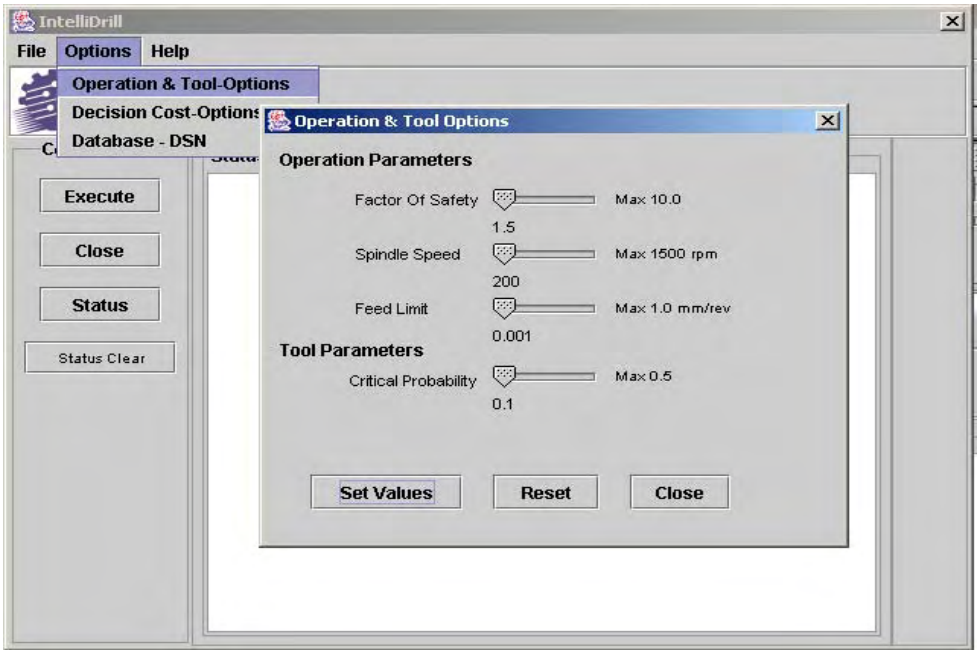

Figure 10. GUI: Operation & Tool Options

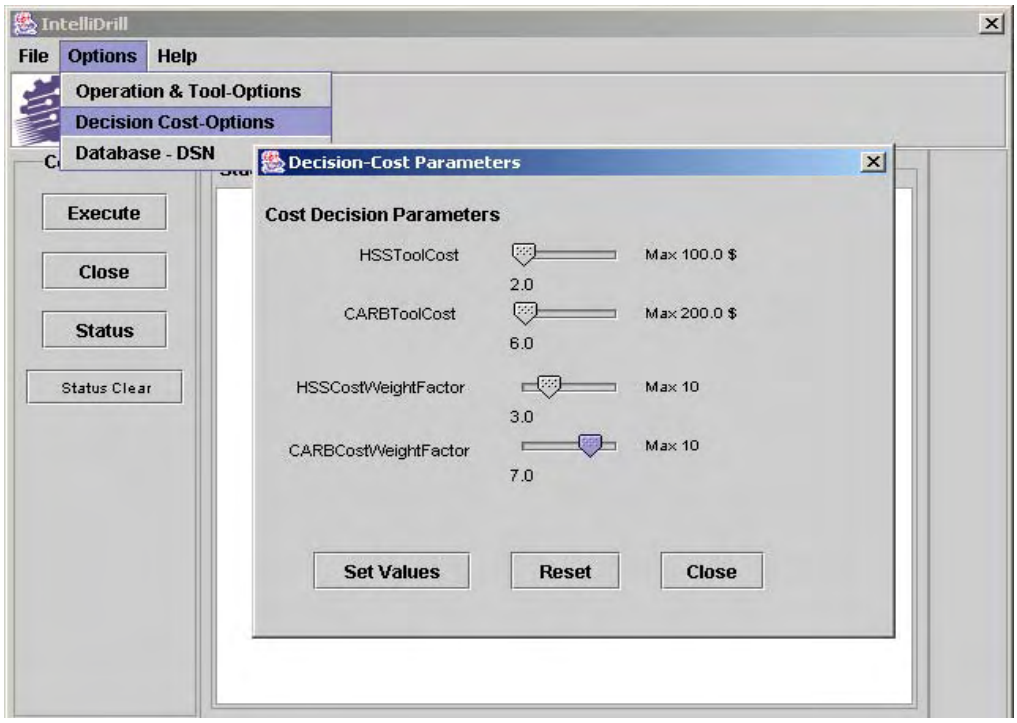

Figure 11. GUI: Cost Options for Decision Making

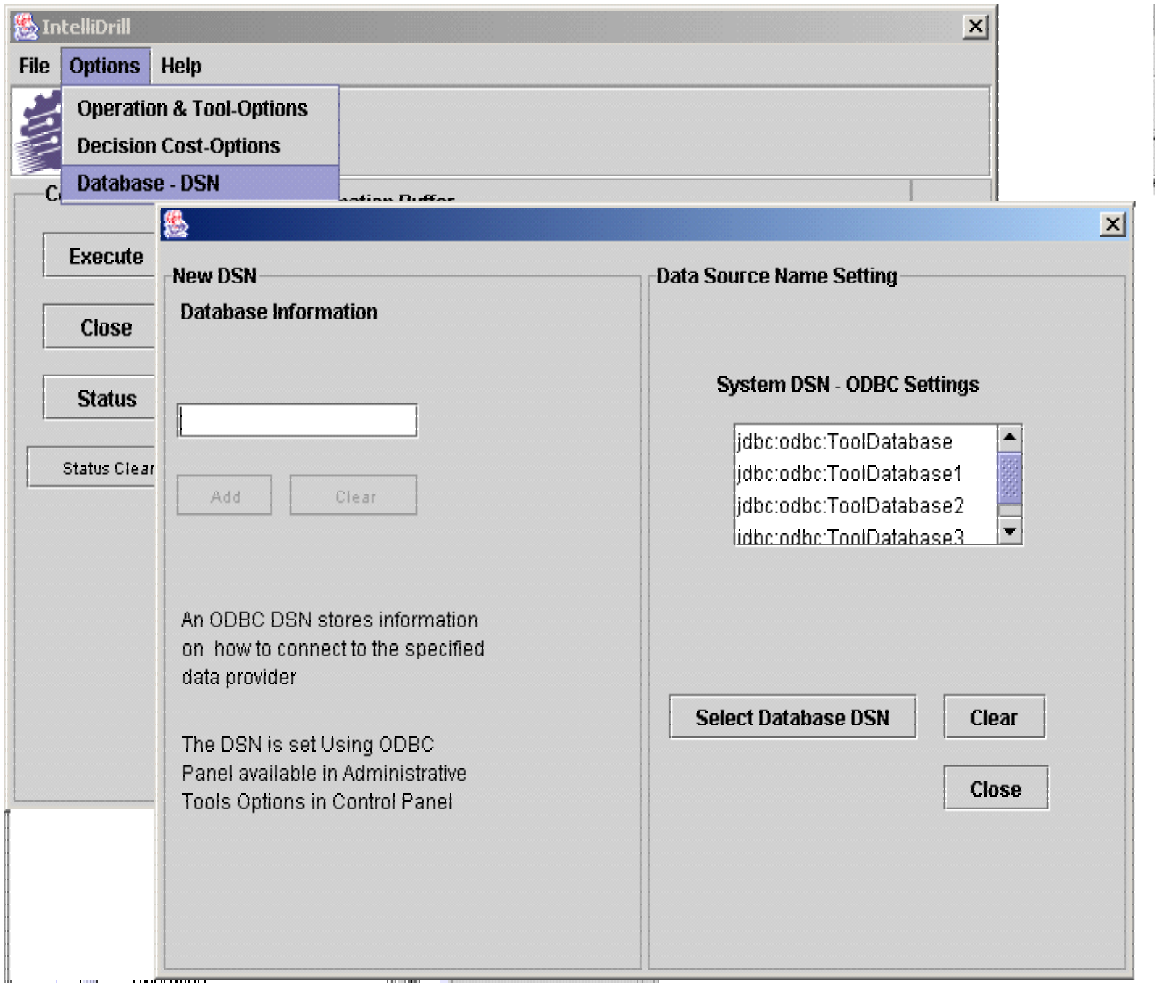

Figure 12. Accessing the Database

4.7 The Database Field

A database provides information regarding tools available, history of tool failures and information about the batch. MS Access has been used as the database for the data definition and data manipulation activities.

4.7.1 ToolList

ToolList database stores information about the tools available for machining. The schema of this database is shown in Figure 13.

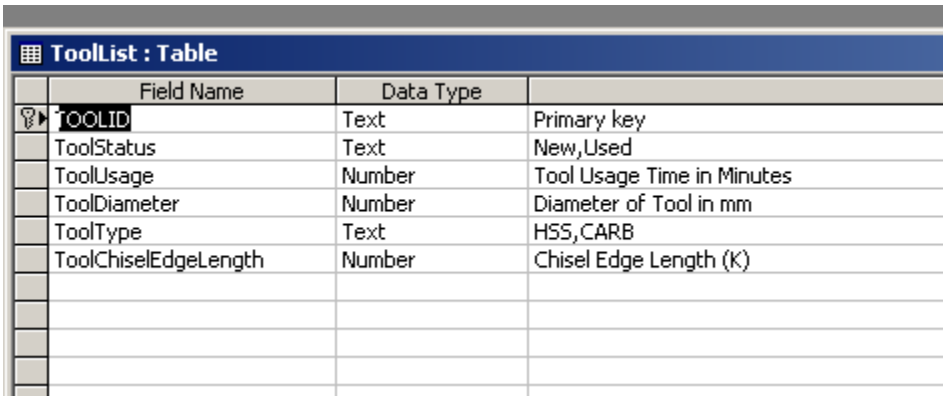

Figure 13. ToolList Database

4.7.2 BatchList

BatchList stores information about the scheduled batches. Each batch is identified using a batch identification number. The schema of this database is shown in Figure 14.

BatchList : Table

	Field Name	Data Type	
🔑	BATCHID	Text	Batch Indetification Number
	BatchSize	Number	Size of the batch
	BatchMaterialHardness	Number	Brinell Hardness Number
	BatchTime	Number	Total Batch Processing Time
	BatchStatus	Text	NEW/OLD/USE
	BatchUnitSetupTime	Number	Unit setup time
▶	BatchUnitMachiningTime	Number	Unit machining time

Figure 14. BatchList Database

4.7.3 ToolFailureList

ToolFailureList records the tool failures. This database stores history of tool failures categorized by the tool type. Schema of this database is shown in Figure 15.

ToolFailureList : Table

	Field Name	Data Type	
🔑▶	TOOLID	Text	Failed Tool
	ToolType	Text	HSS/CARB
	ToolFailTime	Number	Life of Tool in Minutes

Figure 15. ToolFailureList Database

4.7.4 ToolTypeList

This list stores the information about the type of tools being used for an operation. Schema of this database is shown in Figure 16.

ToolTypeList : Table

	Field Name	Data Type	
🔑▶	ToolType	Text	HSS,CARB
	ToolChangeOverTime	Number	Time to Change the Tool in Machine
	ToolUltimateCrushingStress	Number	Units
	ToolUltimateShearingStress	Number	Units
	ToolMeanLife	Number	MeanLife Specified by the Manufacturer

Figure 16. ToolTypeList Database

4.8 Case Analysis: Adaptive Feature of *INTELLIDRILL*

The case analysis presented here displays the adaptive feature of the *INTELLIDRILL* system. A sample case of 2 scenarios is presented in which a HSS tool is considered for operation. Based on the history of tool failures, a tool life of 60 is set. Initially, tool life is set by the management/user. In the first scenario, 15 failure points are observed. (Figure 17). The critical limit is set at 0.667.

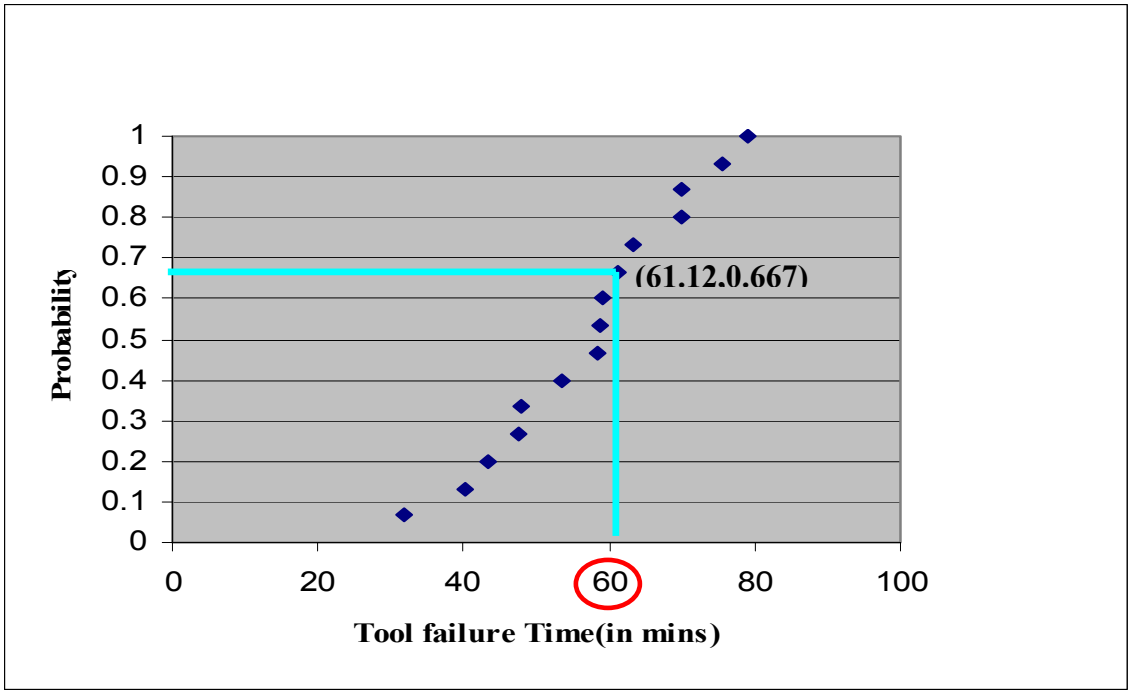

Figure 17. Scenario One – Adaptive Feature of *INTELLIDRILL*

Based on the earlier discussions (Section 3.5), the inverse empirical function yields a permissible time at the maximum acceptable probability of failure which in this case is equivalent to 61.12 minutes. The tool is not available for an operation if it does not fall within this permissible limit. Failure points are recorded in the system and the system recomputes the mean life and the new mean life is applied for the next decision.

In the next operation/tool selection, the system would have adapted to the new mean life and all calculations will be with respect to the new updated mean life.

In the second scenario (Figure 18), 5 more failure points are recorded. The recent tool failures have occurred premature to the allocated machining time (60), bringing down the mean life to 58.88. Thus, in the first case any quantity in the zone of upto 60 would have been acceptable. In this case, the mean life has been already updated by the system and hence 60 is no longer acceptable. This is the adaptive capability of the *INTELLIDRILL* system.

The system has to be initialized with atleast 50 failure points for fairly accurate decision making. The decision making reaches steady state when the number of tool failures in successive decisions is reduced to minimal, which can be considered as noise or just a random occurrence.

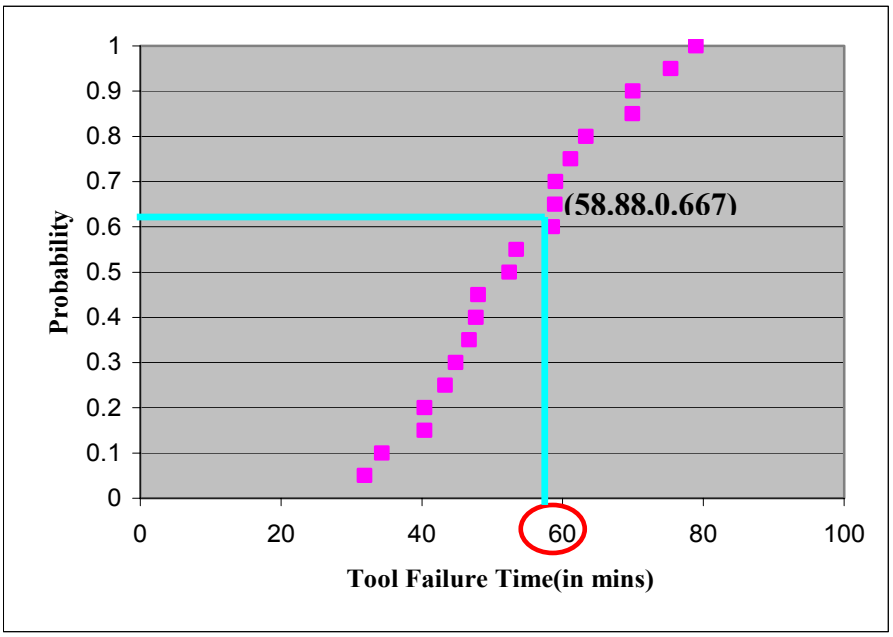

Figure 18. Scenario Two – Adaptive Feature of *INTELLIDRILL*

CHAPTER V

RESULTS AND DISCUSSION

5.1 Experimental Setup

The experimental setup consists of setting up different parameters discussed in the previous sections. The spindle speed limit is set at 1500 rpm. Due to some of the practical constraints during drilling operation, it is always advisable to set the speed not more than 1500 rpm. In practice, a drilling operation is carried on with a low feed rate. A higher feed rate introduces shake/vibration in the tool usually regarded as a noise component. This component can hamper the smooth operation of a process. Thus, a feed rate limit has to be set. This is to ensure a safe allowance for the tool. A feed rate limit of 1.0 mm/rev is chosen. An initial factor of safety as high as 5.0 is chosen. The system will stabilize itself if there is a necessity to lower down the FOS (step down value of 0.1). The iterative procedure ensures that unless an optimal FOS is reached, the system keeps decreasing the FOS. *INTELLIDRILL* will not allow an operation to be performed if the value of the FOS reaches below the threshold value. A decision is made stating that the operation is infeasible. The present system has the threshold value equal to 1.5. *INTELLIDRILL* can accommodate FOS as high as 10.0. A critical availability limit of 0.333 is chosen. This is from a company's perspective. If the company is willing to take further risk, then it can have a lower critical limit. It is a management decision to set the critical limit. If the management is revisiting a problem then, based on previous experience, they can set a critical limit of their own.

In the ToolList database (Figure 13), ToolDiameter is taken as 10 mm for both the tools and ChiselEdgeLength (k) for both the tools is taken as 1. In the BatchList

database (Figure 14), BatchSize for the scenarios are updated, the BHN for the material is a standard entity that varies from material to material. For Aluminum, the BHN is 99-101 whereas for commercially pure beryllium, it is 350. The Batch Unit setup time is 1.0.

In the ToolType list database(Figure 16), the fields: Tool ultimate shearing and crushing stresses, BHN(Brinell hardness number) for both the tools are updated based on the AISI (American Iron and Steel Institute) specification. Tool mean life is to be specified by the user. For this experimental run, the tool mean life for HSS is assumed to be 60 minutes whereas for a carbide tool, it is assumed to be 300 minutes.

The TRE-M has the ToolCostAnalysis function that takes into account the unit cost of both HSS and Carbide tools. A tooling cost of $7 for HSS and $125 for Carbide has been assumed (Gordon and Littleson July 2003). A carbide drill cuts 2.5 times faster than a HSS drill and lasts 5 times as long, but it is also nearly 18 times more expensive (Gordon and Littleson July 2003). Based on all these factors, a weight factor of 5 for HSS and 1 for carbide has been setup in this experimental run. The weight factor takes into account the unit cost of tool, the number of tools of each kind and factor of safety for an operation and computes a simple cost weight factor ratio (Carbide/HSS) to effect a decision. *INTELLIDRILL* will bias the decision towards a Carbide tool if this ratio is greater than or equal to 1.5. It is an individual's perspective to bias either on a HSS or a Carbide tool. If maximum tools present in the tool inventory of a company are HSS then, the management strategy is to bias inadvertently towards HSS. If the management is willing to pay more for Carbide tools for the amount of job they can get done as quickly as possible then, the bias will be more towards a Carbide tool. Setting the delimiters for parameters such as critical limit, weight factors and cost are upto the discrepancy of an individual/management.

The value of the parameters so far discussed is summarized below:

Initial Factor of Safety	**5.0**
Spindle Speed Limit (rpm)	**1500.0**
Feed Rate Limit (mm/rev)	**1.0**
Critical Availability Probability	**0.3333**
Unit High Speed Steel Tool Cost	**$7**
Unit Carbide Tool Cost	**$125**
Weight Factor of High Speed Tool	**5.0**
Weight Factor of Carbide Tool	**1.0**

5.2 Scenario 1: Drilling a Batch of Aluminum

The material considered for drilling is Aluminum. The BHN for aluminum is recorded into the database as 100. A batch size of 40 is entered in the batch size cell (Table 1). The batch time is set to 400 minutes. The remaining parameters are set as mentioned in section 5.1.

BatchList						
BATCHID	**Batch Size**	**Batch Material Hardness**	**Batch Time**	**Batch Status**	**Batch Unit Setup Time**	**Batch Unit Machining Time**
BA101	40	100	400	NEW	1	0

Table 1. Setup for Scenario 1

5.2.1 Result

Operation Information

Operation: OP BA101
Batch Information: ID: BA101
Status: NEW
Size: 40
Brinell Hardness: 100.0
Batch Time: 400.0
Speed: 1499.49 rpm
Feed: 0.078 mm/rev
Material Removal Rate: 1837.22 mm^3/min
Status: ACTIVE Factor of Safety: 5.0

Tools Selected

Tool	Tool ID	Status	Usage(mins)	Diameter(mm)	ToolMaterial
1	TA106	NEW	0.0	10.0	HSS
2	TA111	NEW	0.0	10.0	HSS
3	TA117	NEW	0.0	10.0	HSS
4	TA118	NEW	0.0	10.0	HSS
5	TA119	NEW	0.0	10.0	HSS
6	TA120	NEW	0.0	10.0	HSS

High Speed Steel Tool - Machining

TOOLID: TA106 Machining Time: 60.0
TOOLID: TA111 Machining Time: 60.0
TOOLID: TA117 Machining Time: 60.0

TOOLID: TA118 Machining Time: 60.0

TOOLID: TA119 Machining Time: 60.0

TOOLID: TA120 Machining Time: 60.0

Mathematical Force Model

Axial Thrust Force: 591.72 N

Torque: 34.37 N-mm

Axial Stress: 7.53 N/mm^2

Shear Stress: 2.188 N/mm^2

Machining Feasibility: true

5.2.2 Analysis

The output here displays the feed, speed, the material removal rate, the forces, the tool type and the number of tools of that type needed for the operation. The output indicates 6 different HSS tools are required for the operation. The batch time was entered as 400. The system calculates the total machining time for the operation from the BatchSize, the Batchtime and unit setup. This turns up to be 360 i.e. Total Machining time = 400 – (1*40) = 360. Thus for 360 minutes, the system searches for tools in the database. For this particular operation, the system chose 6 HSS tools which were available in the inventory. All the HSS tools selected are NEW tools. If the operation is HSS feasible then, it is also carbide feasible. The system presented an alternate solution to this scenario choosing 2 carbide tools (Mean life = 300).

Carbide Tool - Machining

TOOLID: TA109 Machining Time: 300.0

TOOLID: TA104 Machining Time: 60.0

The weight factor was 5 for HSS and 1 for Carbide. Based on the discussions in Section 5.1, the value of the cost weight factor was obtained as 0.995. Thus *INTELLIDRILL* chose HSS over carbide even though Carbide was operation feasible.

INTELLIDRILL decisions can be validated with respect to Tool and Manufacturing Engineers Handbook. Aluminum is best machined by a HSS. The speed should be within the permissible limits of 175-400 sfpm (1697-3880 rpm). The feed should be within 0.1778 mm/rev. The speed and feed are 1499.49 rpm and 0.078 mm/rev respectively which are under permissible limits and the tool chosen is also HSS which satisfies all the basic mandatory conditions.

5.3 Scenario 2: Variation in Batch Size (Extension of Scenario 1)

The material considered for drilling is Aluminum. BHN = 100. A batch size of 80 is entered in the batch size cell (Table 2). The batch time is set to 400 minutes. The remaining parameters are set as mentioned in section 5.1.

BatchList						
BATCHID	**Batch Size**	**Batch Material Hardness**	**Batch Time**	**Batch Status**	**Batch Unit Setup Time**	**Batch Unit Machining Time**
BA102	80	100	400	NEW	1	0

Table 2. Setup for Scenario 2

5.3.1 Result

Operation Information

Operation: OP BA102
Batch Information: ID: BA102
Status: NEW
Size: 80
Brinell Hardness: 100.0
Batch Time: 400.0
Speed: 1499.49 rpm
Feed: 0.176 mm/rev
Material Removal Rate: 4161.18 mm^3/min
Status: ACTIVE Factor of Safety: 5.0

Tools Selected

Tool	Tool ID	Status	Usage(mins)	Diameter(mm)	ToolMaterial
1	TA106	NEW	0.0	10.0	HSS
2	TA111	NEW	0.0	10.0	HSS
3	TA117	NEW	0.0	10.0	HSS
4	TA118	NEW	0.0	10.0	HSS
5	TA119	NEW	0.0	10.0	HSS
6	TA122	OLD	30.0	10.0	HSS

High Speed Steel Tool - Machining

TOOLID: TA106 Machining Time: 60.0
TOOLID: TA111 Machining Time: 60.0
TOOLID: TA117 Machining Time: 60.0
TOOLID: TA118 Machining Time: 60.0
TOOLID: TA119 Machining Time: 60.0
TOOLID: TA122 Machining Time: 20.0

Mathematical Force Model

Axial Thrust Force: 1074.81 N
Torque: 66.11 N-mm
Axial Stress: 13.68 N/mm^2
Shear Stress: 4.21 N/mm^2
Machining Feasibility: true

5.3.2 Analysis

The output displays the feed, speed, the material removal rate, the forces, the tool type and the number of tools of that type needed for the operation. The output indicates 6 different HSS tools are required for the operation. The batch time was entered as 400. The system calculates the total machining time for the operation from the batch size, the batch time and unit setup i.e. Total Machining time = 400 – (1*80) = 320 minutes. Thus for 320 minutes, the system searches for tools in the database. For this particular operation, the system chose 5 new tools which were available in the inventory. Instead of using a new tool, the system scans for an old tool in the inventory. The old tool should also have a critical availability probability that is more than 0.333 and the tool usage must be sufficient to fulfill the remaining machining time. Thus the system chose 6

different HSS tools (Mean Life = 60). If the operation is HSS feasible then, it is also carbide feasible. The system presented an alternate solution to this scenario choosing 2 carbide tools (Mean life = 300).

Carbide Tool - Machining

TOOLID: TA109 Machining Time: 300.0

TOOLID: TA104 Machining Time: 20.0

The weight factor was 5 for HSS and 1 for Carbide. Based on the discussions in Section 5.1, the value of the cost weight factor was obtained as 0.995. Thus *INTELLIDRILL* chose HSS over carbide even though Carbide was operation feasible.

As mentioned earlier, the speed should be within the permissible limits of 175-400 sfpm (1697-3880 rpm). The feed should be within 0.1778 mm/rev. The speed and feed are 1499.49 rpm and 0.176 mm/rev respectively. Both the speed and the feed are within permissible limits. The feed rate has doubled leading to an increase in the axial force and the torque. In turn, the stresses have also increased. Comparing Scenario 2 to Scenario 1, it can be noticed that there has been an increase in the batch size without an increase in the machining time. Thus, the system provides an option to machine batch size of either 40 or 80 maintaining the same machining time. It adjusts the unit machining time accordingly and outputs the requisite parameters.

5.4 Scenario 3: Variation in Batch Processing Time

The material considered is Aluminum. BHN = 100. A batch size of 80 is entered in the batch size cell (Table 3). The batch time is not specified in this case. The remaining parameters are set as mentioned in section 5.1.

BatchList						
BATCHID	**Batch Size**	**Batch Material Hardness**	**Batch Time**	**Batch Status**	**Batch Unit Setup Time**	**Batch Unit Machining Time**
BA103	80	100	0	NEW	1	0

Table 3. Setup for Scenario 3

5.4.1 Result

Operation Information

Operation: OP BA103
Batch Information: ID: BA103
Status: NEW
Size: 80
Brinell hardness: 100.0
Batch Time: 0.0
Speed: 1499.99 rpm
Feed: 0.88 mm/rev
Material Removal Rate: 20944.22 mm^3/min
Status: ACTIVE Factor of Safety: 5.0

Tools Selected

Tool	Tool ID	Status	Usage(mins)	Diameter(mm)	ToolMaterial
1	TA112	NEW	0.0	10.0	CARB

CARBIDE Tool – Machining -Minimum Processing Time Case

TOOLID: TA112 Machining Time: 60.0

Mathematical Force Model

Axial Thrust Force: 3732.65 N

Torque: 240.73 N-mm

Axial Stress: 47.53 N/mm^2

Shear Stress: 15.33 N/mm^2

Mechanical Stress Analysis

Machining Feasibility: true

5.4.2 Analysis

The output displays the feed, speed, the material removal rate, the forces, the tool type and the number of tools of that type needed for the operation. The output indicates 1 Carbide tool is required for the operation. The batch time was entered as 0. This is a minimum processing time case. In such a case, *INTELLIDRILL* has to calculate the minimum feasible machining time in which the given batch can be machined. The unit setup time is assumed to be known. The initial batch item is allotted to be just greater than batch total setup time.

This ensures non-zero positive machining time. The MRR is computed for this initial batch processing time. The feed-speed combination to attain the MRR is determined and this combination should be within the permissible limits. If not the machining time assumed is not sufficient to machine the given batch and is stepped up to the next value. The minimum time is estimated at a point where feed, speed and the stresses generated are within the permissible limits.

The estimation process is depicted in Figure 19. The initial total machining time is taken to be 10 minutes. This is linearly increased in steps of 10 until a point of feasibility is reached. After 5 iterations, the system calculates the feasible total machining time which is equal to 60 minutes. Until 60 minutes, the desired MRR was greater than the maximum allowed MRR, hence making the machining infeasible. The reduction of MRR is hyperbolic as it is inversely proportional to the total machining time. The system also accounts for tool change over time and reduces the machining time to 55 minutes (tool changeover time for Carbide = 5 minutes) and the final MRR estimated lies below the maximum allowed MRR.

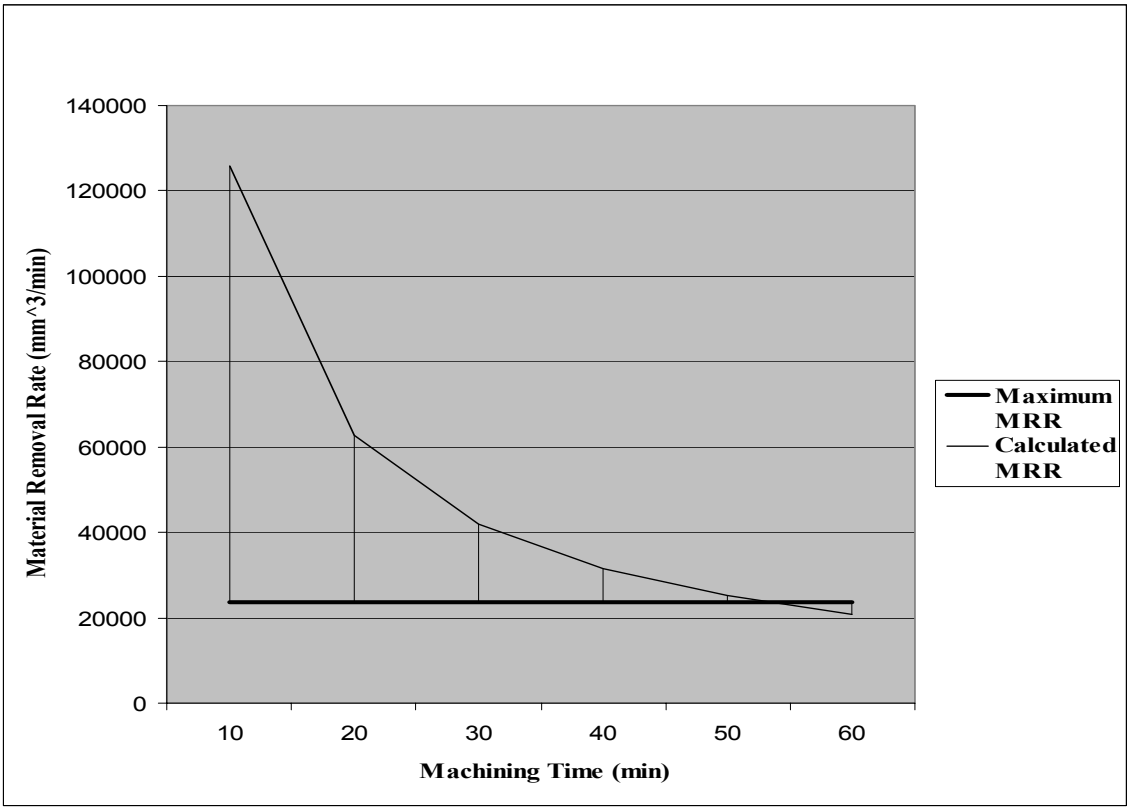

Figure 19. Graph Depicting Minimum Processing Time Case

As mentioned earlier, the speed should be within the permissible limits of 175-400 sfpm (1697-3880 rpm). The feed should be within 0.1778 mm/rev. The speed and feed are 1499.99 rpm and 0.88 mm/rev respectively. The speed is within permissible limits. The feed rate has largely increased leading to an increase in the axial force, the torque and the stresses. Comparing Scenario 3 to Scenario 2, it can be noticed that the batch size remains the same but the machining time has been reduced to 60 minutes (Scenario 2 = 320 minutes). Thus, the machining time has been reduced by a little more than 5 times the value of the feed in Scenario 2. In the minimum processing time case, the emphasis is on expediting the job process as soon as possible rather than saving the tooling costs. In order to effect this process and machine the batch within the given 60 minutes, *INTELLIDRILL* increases the feed to a value that is almost 5 times the value of the feed in Scenario 2. The system provides an option to machine a batch size of 80

anywhere between 60-320 minutes. Thus, there is a compromise between the tooling costs and the machining time.

5.5 Nature of Speed, Feed and MRR in Relation to Batch Size

From the above experimental runs, graphs have been generated to analyze the type of variation in speed, feed and MRR with respect to the batch size (Figures 20, 21, 22). The batch size linearly increases from 10 to 100. In order to accommodate the linear increase in MRR, the feed is varied by keeping the speed a constant. From the graphs, it can be analyzed that there is a linear increase in the MRR for an increase in the batch size. There is a linear increase in the feed as the batch size increases but the speed is a constant for any increase in the batch size.

MRR is a linear combination of speed and feed. In all the experimental runs, the speed is kept a constant. To maintain the MRR to be within permissible limits, either one of the parameters (speed or feed) has to be kept constant. A similar experimental run can be performed keeping the feed constant and increasing the speed linearly.

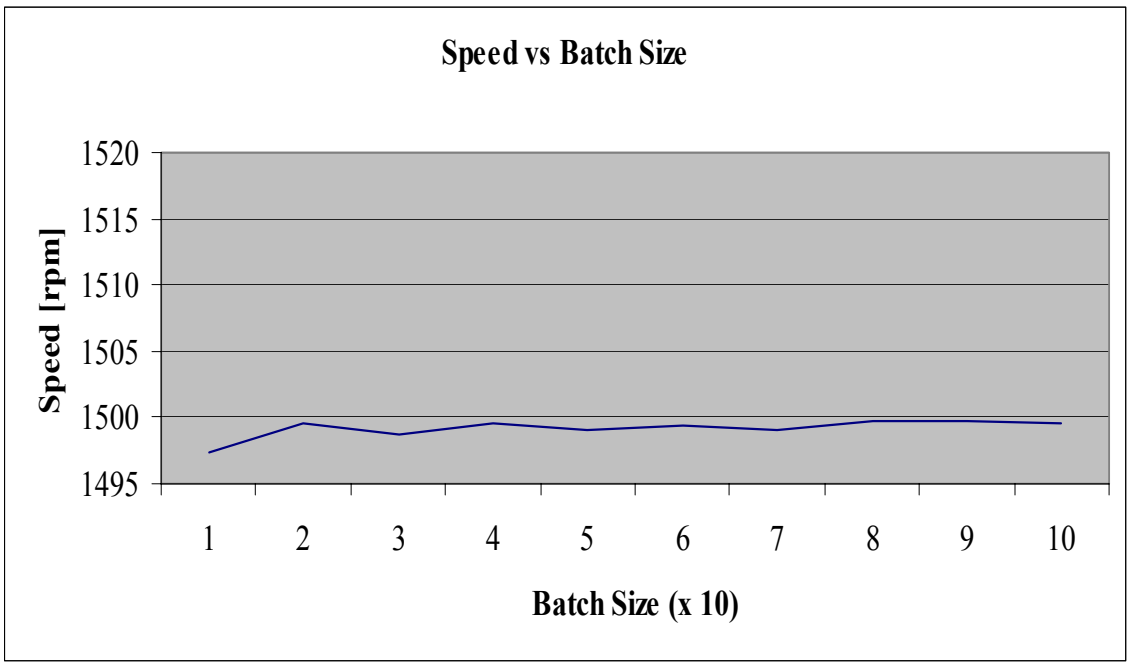

Figure 20. Graph Depicting Speed vs Batch Size

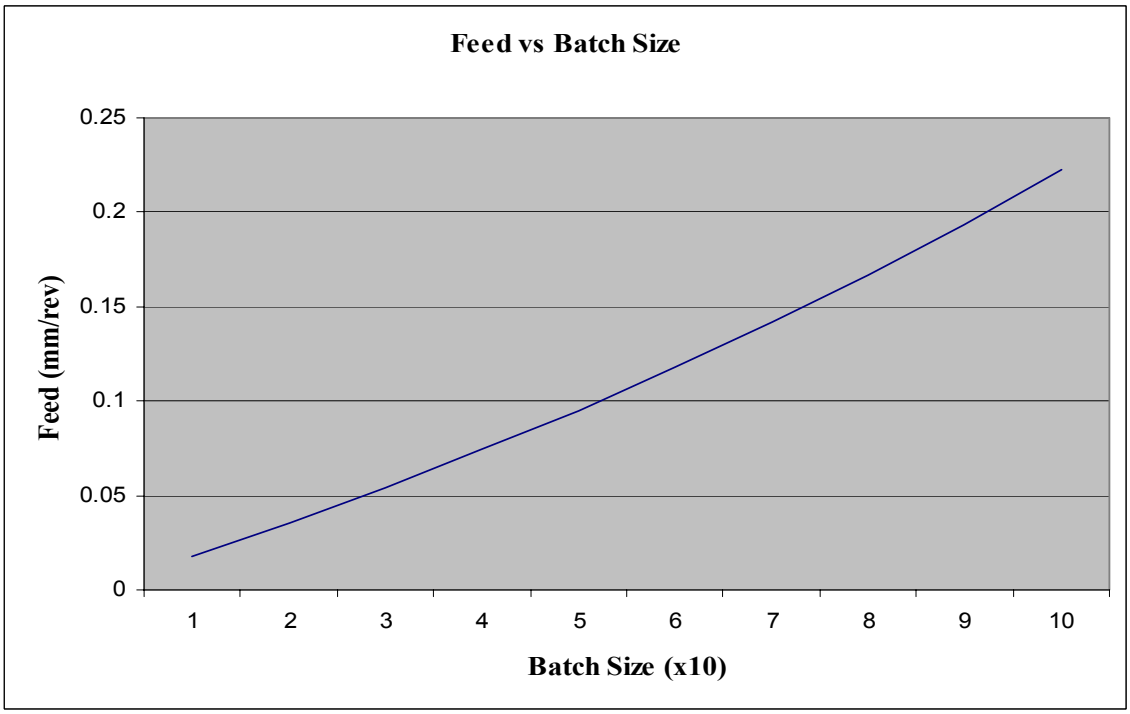

Figure 21. Graph Depicting Feed vs Batch Size

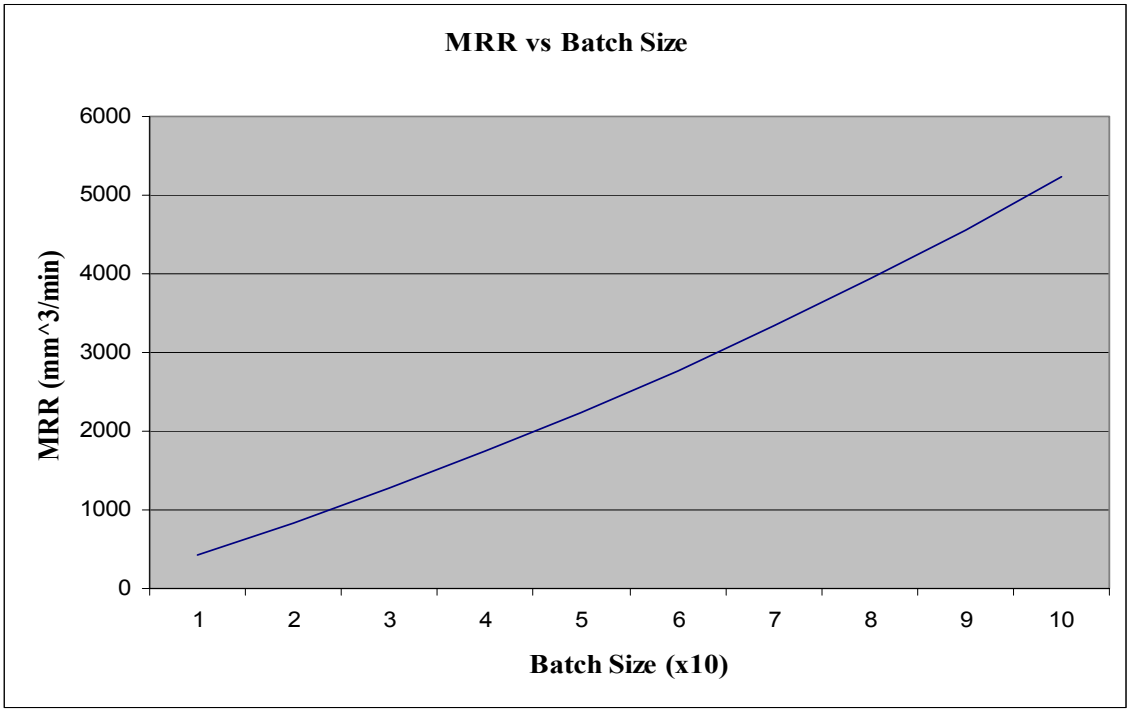

Figure 22. Graph Depicting MRR vs Batch Size

CHAPTER VI

CONCLUSIONS AND FUTURE SCOPE

6.1 Contribution of the Thesis

The thesis presents a methodology to design an adaptive decision making system for a FMS. *INTELLIDRILL* is capable of handling real time decisions for a drilling center in a FMS. *INTELLIDRILL* provides a better tooling strategy for the given batch time. It conveniently adapts to scenarios where the batch processing time is not provided. Such a scenario which is regarded as a minimum processing time case (discussed in Chapter IV) can be handled by the system and the minimum possible time to machine the batch is computed. A feasible tooling strategy for the minimum batch time is then estimated.

The tool failure during an operation is recorded and is stored in the Tool Failure List. In the next iteration/decision setup, these data points are immediately adapted. The decision making approach is modular in nature. The thesis attempts to put together a framework to accommodate mathematical and reliability models owing to the flexible nature of *INTELLIDRILL*. The adaptive decision making strategy helps in extending tool life and reducing bottlenecks. *INTELLIDRILL* decisions could lead to significant savings in tooling costs.

6.2 Future Scope of Study

Future work on this topic can be suggested in the following areas:

1. The ideology applied to this operational model can be further extended to other machining operations like Turning, Milling etc. An adaptive decision making system can serve as backend intelligence for milling machines and other similar machines in a FMS layout.
2. At present, only two types of tools have been considered for designing the system. In future, several kinds of tools can be introduced in the system.
3. Integrating the GUI with visualization aid to understand the impact of the decision process.
4. A more powerful statistical distribution that can closely mimic tool failures can be worked out in the future.
5. The type of incoming material in a batch is assumed to have uniform thickness. In future, decisions related to materials with non uniform thickness should be considered.

REFERENCES

Adam, E.E., Jr., Ebert, R.J., 1999, *Production and Operations Management, 5th edn,* (New Delhi: Prentice- Hall).

DeGarmo Paul, E., TempleBlack, J., Kohser, R.A., 1984, *Materials and Processes in Manufacturing*, 6th edn, (New York: Macmillan).

Duffie, N.A., Chitturi, R., Mou, J.I., 1988, Fault-tolerant Heterarchical Control of Heterogeneous Manufacturing System Entities*,* 315-319, Department of Mechanical Engineering, University of Wisconsin-Madison, Madison, Wisconsin, USA.

Gordon, K., Littleson, L., July 2003, Modern Machine Shop,http://www.mmsonline.com

Hindustan Machine Tools, 1997, *Production Technology Handbook* (New Delhi: Tata Mcgraw-Hill).

Larin, D.J., January 1989, Cell Control: What We Have, What We'll Need, *Manufacturing Engineering*, 41-54.

Machinability Data Center, 1972, *Machining Data Handbook,* 2nd edn (Cincinnati, Metcut Research Associates Inc).

Metcut Research Associates Inc, 1966, *Machining Data Handbook*, (Cincinnati, Metcut Research Associates Inc).

Mills, B., Redford, A.H., 1983, *Machinability of Engineering Materials* (New York: Applied Science).

Sandvik Coromant, 1996, *Modern Metal Cutting - A Practical Handbook*: 1st North American edn, (Fair Lawn, NJ: Sandvik Coromant).

Stephenson, D.A., Agapiou, J.S., 1997, *Metal Cutting Theory and Practice* (New York: Marcel Dekker Inc).

Society of Manufacturing Engineers, 1976, Tool *and Manufacturing Engineers Handbook*, (New York: McGraw-Hill).

Trent, E.M., 1984, *Metal Cutting,* 2nd edn, (Boston: Butterworths).

Trent, E.M., and Wright P.K., 1999, *Metal Cutting,* 4th edn, (Boston: Butterworth-Heinemann).

Tung, L.F., Lin, L., Nagi, R., 1999, Multiple-objective Scheduling for the Hierarchical Control of Flexible Manufacturing Cells, *International Journal of Flexible Manufacturing Systems*, **11**(4), 379-409.

Upton, D.M., 1992, A Flexible Structure for Computer-Controlled Manufacturing Systems, *Manufacturing Review* **5** (1), 58-74.

APPENDIX A

API DOCUMENTATION FOR CLASS – TOOL

Class Tool

java.lang.Object

└ **Tool**

All Implemented Interfaces:

_Tool

---

public class **Tool**

extends java.lang.Object

implements _Tool

Tool Class deals with all Tool Management Functions

Field Summary

protected static double	**CriticalLimit** Critical Probability of Tool Availability

Method Summary

boolean	**checkToolAvailability**() Checks the availability of the tool for the operation
void	**copyInformation**(Tool _Tool)

	Copies Information from specified Tool
double	**F**(double _Time)
	Computes the Probability of Failure for a specified Machining Time
protected void	**finalize**()
	finalizes the Tool Class
double	**Finv**(double _Probability)
	Computes the Time in which the given tool fails with the specified probability
double	**getToolChangeTime**()
	returns Tool Change Over Time
double	**getToolChiselEdgeLength**()
	returns the Tool Chisel Edge Length
double	**getToolDiameter**()
	returns Tool Diameter
java.lang.String	**getToolID**()
	returns Tool ID
double	**getToolMachiningTime**()
	returns Tool Machining Time
double	**getToolMeanLife**()
	returns Mean Life of the Tool
java.lang.String	**getToolStatus**()
	returns Tool Status
java.lang.String	**getToolType**()
	returns Tool Type
double	**getToolUltimateCrushingStress**()
	returns Tool Material Ultimate Crushing Stress
double	**getToolUltimateShearingStress**()
	returns Tool Material Ultimate Shearing Stress

double	**getToolUsage**() returns time of Tool Usage
void	**loadNewTool**(java.lang.String _ToolType) Loads a new tool from the tool database with the specified String _ToolType
void	**loadTool**(double _ToolTime, java.lang.String _ToolType) Loads an old tool from the tool database with the specified arguments double _MachiningTime and String _ToolType
void	**loadTool**(java.lang.String _TOOLID) Loads a tool from the tool database with the specified _TOOLID
void	**printToolInformation**() Prints the Tool Information
void	**setToolMachiningTime**(double _ToolMachiningTime) Sets the Machining Time allocated for the Tool
void	**setToolUsage**(double _ToolUseTime) Increments the usage of the Tool by the specified value.
java.lang.String	**ToolStatusInformation**() Prints the Tool Status Information
void	**unloadTool**() unloads a given tool and updates the Tool Database

Methods inherited from class java.lang.Object

clone, equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

CriticalLimit

protected static double **CriticalLimit**

Critical Probability of Tool Availability

Method Detail

loadNewTool

public void **loadNewTool**(java.lang.String _ToolType)

Loads a new tool from the tool database with the specified String _ToolType

Parameters:

_ToolType – Tool Material Specification

---

loadTool

public void **loadTool**(double _ToolTime,
java.lang.String _ToolType)

Loads an old tool from the tool database with the specified arguments double _MachiningTime and String _ToolType

Parameters:

_ToolTime – Machining Time for the Tool

_ToolType – Tool Material Specification

---

loadTool

public void **loadTool**(java.lang.String _TOOLID)

Loads a tool from the tool database with the specified _TOOLID

Parameters:

_TOOLID – Tool Identification Number

---

unloadTool

public void **unloadTool**()

unloads a given tool and updates the Tool Database

F

public double **F**(double _Time)

Computes the Probability of Failure for a specified Machining Time

Parameters:

_Time – Machining Time of Operation

Returns:

reutns the Failure Probability

Finv

public double **Finv**(double _Probability)

Computes the Time in which the given tool fails with the specified probability

Parameters:

_Probability – Probability of Tool Failure

Returns:

Time of Failure

---

checkToolAvailability

public oolean **checkToolAvailability**()

Checks the availability of the tool for the operation

Returns:

Availability of the Tool

---

printToolInformation

public void **printToolInformation**()

Prints the Tool Information

ToolStatusInformation

public java.lang.String **ToolStatusInformation**()

Prints the Tool Status Information

Returns:

Tool Information

---

setToolUsage

public void **setToolUsage**(double _ToolUseTime)

Increments the usage of the Tool by the specified value

Parameters:

_ToolUseTime – Time Tool has been used for the current Operation

---

setToolMachiningTime

public void **setToolMachiningTime**(double _ToolMachiningTime)

Sets the Machining Time allocated for the Tool

Parameters:

_ToolMachiningTime – Tool Machining Time

---

getToolID

public java.lang.String **getToolID**()

returns Tool ID

Returns:

Tool ID

---

getToolStatus

public java.lang.String **getToolStatus**()

returns Tool Status

Returns:

Tool Status

---

getToolUsage

public double **getToolUsage**()

returns time of Tool Usage

Returns:

Tool Usage

---

getToolDiameter

public double **getToolDiameter**()

returns Tool Diameter

Returns:

Tool Diameter

---

getToolType

public java.lang.String **getToolType**()

returns Tool Type

Returns:

Tool Type

---

getToolChangeTime

public double **getToolChangeTime**()

returns Tool Change Over Time

Returns:

Tool Change Over Time

---

getToolUltimateCrushingStress

public double **getToolUltimateCrushingStress**()

returns Tool Material Ultimate Crushing Stress

Returns:

Ultimate Crushing Stress

---

getToolUltimateShearingStress

public double **getToolUltimateShearingStress**()

returns Tool Material Ultimate Shearing Stress

Returns: Ultimate Shearing Stress

---

getToolMeanLife

public double **getToolMeanLife**()

returns Mean Life of the Tool

Returns:

Tool Mean Life

---

getToolChiselEdgeLength

public double **getToolChiselEdgeLength**()

returns the Tool Chisel Edge Length

Returns:

Tool Chisel Edge Length

---

getToolMachiningTime

public double **getToolMachiningTime**()

returns Tool Machining Time

Returns:

Tool Machining Time

---

finalize

protected void **finalize**()

finalizes the Tool Class

---

copyInformation

public void **copyInformation**(Tool _Tool)

Copies Information from specified Tool

Parameters:

_Tool – Original Tool

APPENDIX B

API DOCUMENTATION FOR CLASS – BATCH

Class Batch

java.lang.Object

└ **Batch**

---

public class **Batch**

extends java.lang.Object

Batch Class deals with all Batch Management Functions

Method Summary

void	**adjustBatchTotalMachiningTime**(double _Time) Decrements the Total Machining Time of Batch by specified time
void	**adjustBatchUnitMachiningTime**() Adjusts the Unit Machining Time
java.lang.String	**BatchStatusInformation**() Prints Batch Information to a String
double	**getBatchHoleHeight**() returns Depth of Hole to be drilled for a Job
java.lang.String	**getBatchID**() returns Batch ID
double	**getBatchMaterialHardness**() returns Batch's Material Hardness (BHN)

int	**getBatchSize**()
	returns the Size of the Batch
java.lang.String	**getBatchStatus**()
	returns Batch Status
double	**getBatchTime**()
	returns Batch Processing Time
double	**getBatchTotalMachiningTime**()
	returns Total Machining Time for a Batch
double	**getBatchUnitMachiningTime**()
	returns Unit Machining Time of a Job
double	**getBatchUnitSetupTime**()
	returns Unit Setup Time of a Job
void	**loadNewBatch**()
	Loads a New Batch from the Batch Database
boolean	**prepareBatch**()
	Calculates the Batch Processing Time, Unit Processing Time and Unit Machining Time
void	**printBatchInformation**()
	Prints Batch Information
void	**setBatchProcessingTime**(double _BatchTime)
	sets the Batch Time to the specified value
void	**stepUpBatchTime**()
	Steps up the Batch Time by increment stepValue
void	**unloadBatch**()
	Unloads a Batch into the Batch Database

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Method Detail

loadNewBatch

public void **loadNewBatch**()

Loads a New Batch from the Batch Database

unloadBatch

public void **unloadBatch**()

Unloads a Batch into the Batch Database

---

prepareBatch

public boolean **prepareBatch**()

Calculates the Batch Processing Time, Unit Processing Time and Unit Machining Time

Returns:

Case Type

---

printBatchInformation

public void **printBatchInformation**()

Prints Batch Information

BatchStatusInformation

public java.lang.String **BatchStatusInformation**()

Prints Batch Information to a String

Returns:

Batch Information

---

getBatchID

public java.lang.String **getBatchID**()

returns Batch ID

Returns:

Batch ID

---

getBatchSize

public int **getBatchSize**()

returns the Size of the Batch

Returns:

Batch Size

---

getBatchMaterialHardness

public double **getBatchMaterialHardness**()

returns Batch's Material Hardness (BHN)

Returns: Material Hardness (BHN)

getBatchTime

public double **getBatchTime**()

returns Batch Processing Time

Returns:

Batch Processing Time (min)

---

getBatchStatus

public java.lang.String **getBatchStatus**()

returns Batch Status

Returns:

Batch Status

---

getBatchUnitSetupTime

public double **getBatchUnitSetupTime**()

returns Unit Setup Time of a Job

Returns:

Unit Setup Time (min)

---

getBatchUnitMachiningTime

public double **getBatchUnitMachiningTime**()

returns Unit Machining Time of a Job

Returns:

Unit Machining Time (min)

---

getBatchTotalMachiningTime

public double **getBatchTotalMachiningTime**()

returns Total Machining Time for a Batch

Returns:

Total Machining Time (min)

getBatchHoleHeight

public double **getBatchHoleHeight**()

returns Depth of Hole to be drilled for a Job

Returns:

Hole Depth

setBatchProcessingTime

public void **setBatchProcessingTime**(double _BatchTime)

sets the Batch Time to the specified value

Parameters:

_BatchTime - Batch Time

stepUpBatchTime

public void **stepUpBatchTime**()

Steps up the Batch Time by increment stepValue

adjustBatchTotalMachiningTime

public void **adjustBatchTotalMachiningTime**(double _Time)

Decrements the Total Machining Time of Batch by specified time

Parameters:

_Time - Decrement Time

adjustBatchUnitMachiningTime

public void **adjustBatchUnitMachiningTime**()

Adjusts the Unit Machining Time

APPENDIX C

API DOCUMENTATION FOR CLASS – OPERATION

Class Operation

java.lang.Object

└ **Operation**

All Implemented Interfaces:

_Operation

---

public class **Operation**

extends java.lang.Object

implements _Operation

Class Deals with Operation (Managing Drilling Operation)

Field Summary

protected double	**_ShearForce** Shear Force (N)
protected double	**_ThrustForce** Thrust Force (N)
Boolean	**BatchTimeSpecified** Condition: Batch Time Specified
protected double	**StepUpFeed** Step Value of Feed Rate (mm/rev)
protected double	**StepUpSpeed** Step Value of Spindle Speed (rpm)

Method Summary

Void	**adjustBatchMachiningTime**(double _Time) Decrememnts the Batch Total Machining Time by specified value and calculates the Unit Machining Time
Boolean	**approximate**(double a,double b) Checks if the 2 Numbers are almost equal with a predefined Threshold
Boolean	**approximate**(double a,double b, double _SpecifiedTolerance) Checks if the 2 Numbers are almost equal with the specified Threshold
Boolean	**calculateMachiningParameters**() Calculates the Machining Parameters(Material Removal Rate, Feed & Speed)
Boolean	**checkMachiningFeasibility**() Checks the Machining Feasibility of Operation
Boolean	**checkMachiningFeasibility**(Tool _preLoadTool) Checks the Machining Feasibility of the Operation with the Specified Tool
Double	**checkTrayAvailability**() Checks the Availability of Tool in the Tray based on Emperical Distribution and computes the Machining Time remaining to be allocated
Void	**clear**() Clears the Content of the Operation

Void	**cloneOperation**(Operation _Operation) Copies the Information from the specified Operation
Void	**emptyTray**() Empties the Content of the Tray
Void	**estimateBatchProcessingTime**() Estimates Batch Processing Time
Void	**estimateFailureTime**(double _Probability) Estimates Failure Time for given probability of failure from Tool Failure Distribution
Boolean	**estimateFeedSpeed**() Estimates the Feed and Speed of Operation based on Material Removal Rate
Double	**estimateInitialBatchTime**() Estimates Initial Batch Processing Time in the Minimum Processing Time-Case
Double	**estimateOperationToolChangeTime**() Estimates the Total Tool Change Over Time for an Operation
Batch	**getBatch**() returns the Batch
Double	**getCrossSectionalArea**() returns the Cross Sectional Area of the Tool
Double	**getDepthOfCut**() returns Depth of Cut
Double	**getFeed**() returns Feed Rate
Double	**getFOS**() returns Factor of Safety of Operation
double	**getMRR**()

	returns Material Removal Rate for Operation
Double	**getMRR**(double _Speed,double _Feed)
	returns the Material Removal Rate for specified Feed and Speed
Double	**getSpeed**()
	returns Spindle Speed
Double	**getThrustForce**()
	returns the Thrust Force [Shaw & Oxford Equation]
Tool	**getTool**()
	returns Tool
Tool	**getTool**(int _ToolNumber)
	returns Tools at Specified Tray Index
Double	**getTorque**()
	returns the Torque [Shaw & Oxford Equation]
java.util.Stack	**getTray**()
	returns the Tray as a Stack
Int	**getTraySize**()
	returns Tray Size
Boolean	**isFeasible**()
	Checks if the Factor of Safety of Operation is greater than the Threshold Factor of Safety
Void	**loadBatch**(boolean createEmpty)
	Loads a new Batch for Operation
Int	**loadTray**(double _MachiningTime)
	Loads the Tray with Tools for the specified Machining Time
int	**loadTray**(double _MachiningTime, java.lang.String _ToolType)

	Loads the Tray with specified Tool Type for specified Machining Time
Int	**loadTray**(java.lang.String _ToolType)
	Loads the Tray with Specified Tool Type
java.lang.String	**OperationStatusInformation**()
	Prints the Operation Information to the specified String
Void	**printOperationInformation**()
	Prints the Operation Information
Void	**printToolTray**()
	Prints the Tool Information of the Tool Tray
Void	**printTrayMachiningSetup**()
	Prints the Tray Machining Setup
Int	**reloadTray**(double _MachiningTime)
	Reloads the Tray with Tools for specified Remaining Machining Time
Void	**setBatchTime**(double _BatchTime)
	Sets the Batch Time for _Batch
Void	**setDepthOfCut**(double _DepthOfCut)
	Sets the Depth of Cut by specified value
Void	**setFeed**(double _Feed)
	Sets the Feed Rate by specified value
Void	**setFOS**(double FOS)
	Sets the Factor of Safety by specified value
Void	**setOperationID**()
	Sets the Operation ID
Void	**setSpeed**(double _Speed)
	Sets the Spindle Speed by specified value
void	**setStatus**(java.lang.String _Status)

	Sets the Status of Operation by specified value
Void	**setTool**(Tool _setTool)
	Sets the _Tool by the specified Tool
Void	**stepDownFOS**()
	Decrements the Factor of Safety
Void	**stepUpBatchTime**()
	Steps up Batch Time of _Batch
java.lang.String	**TrayMachiningSetupStatusInformation**()
	Prints the Tray Machining Setup to the String
java.lang.String	**TrayStatusInformation**()
	Prints the Tool Information of the Tray to String

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

_ThrustForce

protected double **_ThrustForce**

Thrust Force (N)

_ShearForce

protected double **_ShearForce**

Shear Force (N)

StepUpSpeed

protected double **StepUpSpeed**

Step Value of Spindle Speed (rpm)

StepUpFeed

protected double **StepUpFeed**

Step Value of Feed Rate (mm/rev)

BatchTimeSpecified

public boolean **BatchTimeSpecified**

Condition: Batch Time Specified

Method Detail

cloneOperation

public void **cloneOperation**(Operation _Operation)

Copies the Information from the specified Operation

Parameters:

_Operation - Original Operation

---

loadBatch

public void **loadBatch**(boolean createEmpty)

Loads a new Batch for Operation

Parameters:

createEmpty - Create Empty

---

estimateBatchProcessingTime

public void **estimateBatchProcessingTime**()

Estimates Batch Processing Time

---

loadTray

public int **loadTray**(java.lang.String _ToolType)

Loads the Tray with Specified Tool Type

Parameters:

_ToolType - Tool Type to be Loaded

Returns:

Number of Tools Loaded

---

loadTray

public int **loadTray**(double _MachiningTime)

Loads the Tray with Tools for the specified Machining Time

Parameters:

_MachiningTime - Machining Time

Returns:

Number of Tools Loaded

---

loadTray

public int **loadTray**(double _MachiningTime,
java.lang.String _ToolType)

Loads the Tray with specified Tool Type for specified Machining Time

Parameters:

_MachiningTime - Machining Time (min)

_ToolType - Tool TYpe

Returns:

No. of Tools Loaded

---

reloadTray

public int **reloadTray**(double _MachiningTime)

Reloads the Tray with Tools for specified Remaining Machining Time

Parameters:

_MachiningTime - Remaining Machining Time(min)

Returns:

Tray Size

---

approximate

public boolean **approximate**(double a,
double b)

Checks if the 2 Numbers are almost equal with a predefined Threshold

Parameters:

a - Number1

b - Number 2

Returns:

Check Approximate

---

approximate

public boolean **approximate**(double a,
double b,
double _SpecifiedTolerance)

Checks if the 2 Numbers are almost equal with the specified Threshold

Parameters:

a - Number 1

b - Number 2

_SpecifiedTolerance - Tolerance

Returns:

Check Approximate

---

estimateFeedSpeed

public boolean **estimateFeedSpeed**()

Estimates the Feed and Speed of Operation based on Material Removal Rate

Returns:

Estimated

---

estimateFailureTime

public void **estimateFailureTime**(double _Probability)

Estimates Failure Time for given probability of failure from Tool Failure Distribution

Parameters:

_Probability - Probability of Failure

---

calculateMachiningParameters

public boolean **calculateMachiningParameters**()

Calculates the Machining Parameters(Material Removal Rate, Feed & Speed)

Returns:

isEstimated

---

estimateOperationToolChangeTime

public double **estimateOperationToolChangeTime**()

Estimates the Total Tool Change Over Time for an Operation

Returns:

Total Operation Tool Change Over Time (min)

---

estimateInitialBatchTime

public double **estimateInitialBatchTime**()

Estimates Initial Batch Processing Time in the Minimum Processing Time-Case

Returns:

Batch Processing Time (min)

---

adjustBatchMachiningTime

public void **adjustBatchMachiningTime**(double _Time)

Decrememnts the Batch Total Machining Time by specified value and calculates the Unit Machining Time

Parameters:

_Time - Decrement Time (min)

---

isFeasible

public boolean **isFeasible**()

Checks if the Factor of Safety of Operation is greater than the Threshold Factor of Safety

Returns:

isFeasible

---

printToolTray

public void **printToolTray**()

Prints the Tool Information of the Tool Tray

TrayStatusInformation

public java.lang.String **TrayStatusInformation**()

Prints the Tool Information of the Tray to String

Returns:

Tray Information

---

printTrayMachiningSetup

public void **printTrayMachiningSetup**()

Prints the Tray Machining Setup

TrayMachiningSetupStatusInformation

public java.lang.String **TrayMachiningSetupStatusInformation**()

Prints the Tray Machining Setup to the String

Returns:

Tray Machining Setup

---

emptyTray

public void **emptyTray**()

Empties the Content of the Tray

---

clear

public void **clear**()

Clears the Content of the Operation

---

getTraySize

public int **getTraySize**()

returns Tray Size

Returns:

Tray Size

---

getSpeed

public double **getSpeed**()

returns Spindle Speed

Returns:

Spindle Speed (rpm)

---

getFeed

public double **getFeed**()

returns Feed Rate

Returns:

Feed Rate (mm/rev)

---

getDepthOfCut

public double **getDepthOfCut**()

returns Depth of Cut

Returns:

Depth of Cut (mm)

---

getFOS

public double **getFOS**()

returns Factor of Safety of Operation

Returns:

Factor of Safety

---

getMRR

public double **getMRR**()

returns Material Removal Rate for Operation

Returns:

Material Removal Rate (mm^3/min)

---

getMRR

public double **getMRR**(double _Speed,
double _Feed)

returns the Material Removal Rate for specified Feed and Speed

Parameters:

_Speed - Spindle Speed (rpm)

_Feed - Feed Rate (mm/rev)

Returns:

Material Removal Rate (mm^3/min)

---

getCrossSectionalArea

public double **getCrossSectionalArea**()

returns the Cross Sectional Area of the Tool

Returns:

Cross Sectional Area (mm^2);

---

getThrustForce

public double **getThrustForce**()

returns the Thrust Force [Shaw & Oxford Equation]

Specified by:

getThrustForce in interface _Operation

Returns:

Thrust Force (N)

---

getTorque

public double **getTorque**()

returns the Torque [Shaw & Oxford Equation]

Specified by:

getTorque in interface _Operation

Returns:

Torque (N-mm)

---

checkMachiningFeasibility

public boolean **checkMachiningFeasibility**()

Checks the Machining Feasibility of Operation

Returns:

Machining Feasibility

---

checkMachiningFeasibility

public boolean **checkMachiningFeasibility**(Tool _preLoadTool)

Checks the Machining Feasibility of the Operation with the Specified Tool

Parameters:

_preLoadTool - Pre Load Tool

Returns:

Machining Feasible

---

checkTrayAvailability

public double **checkTrayAvailability**()

Checks the Availability of Tool in the Tray based on Emperical Distribution and computes the Machining Time remaining to be allocated

Returns: Remaining Machining Time (min)

---

getTool

public Tool **getTool**()

returns Tool

Returns:

_Tool

---

getTool

public Tool **getTool**(int _ToolNumber)

returns Tools at Specified Tray Index

Parameters:

_ToolNumber - Tray Index/ Tool Number

Returns:

Tray[i]

---

getBatch

public Batch **getBatch**()

returns the Batch

Returns:

_Batch

---

getTray

public java.util.Stack **getTray**()

returns the Tray as a Stack

Returns:

Tray

---

printOperationInformation

public void **printOperationInformation**()

Prints the Operation Information

OperationStatusInformation

public java.lang.String **OperationStatusInformation**()

Prints the Operation Information to the specified String

Returns:

Operation Information

---

setOperationID

public void **setOperationID**()

Sets the Operation ID

---

setSpeed

public void **setSpeed**(double _Speed)

Sets the Spindle Speed by specified value

Parameters:

_Speed - Spindle Speed (rpm)

---

setFeed

public void **setFeed**(double _Feed)

Sets the Feed Rate by specified value

Parameters:

_Feed - Feed Rate (mm/rev)

---

setDepthOfCut

public void **setDepthOfCut**(double _DepthOfCut)

Sets the Depth of Cut by specified value

Parameters:

_DepthOfCut - Depth of Cut (mm)

---

setStatus

public void **setStatus**(java.lang.String _Status)

Sets the Status of Operation by specified value

Parameters:

_Status - Status

---

setTool

public void **setTool**(Tool _setTool)

Sets the _Tool by the specified Tool

Parameters:

_setTool - Set Tool

---

setFOS

public void **setFOS**(double FOS)

Sets the Factor of Safety by specified value

Parameters:

FOS - Factor of Safety

---

setBatchTime

public void **setBatchTime**(double _BatchTime)

Sets the Batch Time for _Batch

Parameters:

_BatchTime - Batch Time (min)

---

stepDownFOS

public void **stepDownFOS**()

Decrements the Factor of Safety

stepUpBatchTime

public void **stepUpBatchTime**()

Steps up Batch Time of _Batch

APPENDIX D

API DOCUMENTATION FOR CLASS – *INTELLIDRILL*

Class INTELLIDRILL

java.lang.Object

└ Debug

└ **INTELLIDRILL**

Direct Known Subclasses:

Source

---

public class **INTELLIDRILL**

extends Debug

INTELLIDRILL - Class contains decision making modules

Field Summary

static Operation	**_OperationCARB** Operation with Carbide Machining
static Operation	**_OperationHSS** Operation with High Speed Steel Machining
static boolean	**CARBFeasible** feasibility of _OperationCARB
static Tool	**CarbideTool** Carbide Tool
static Debug	**d** Debug Console

static boolean	**ExceedsLimit** flags if MaterialRemovalRate is greater than Maximum Allowed Value
static boolean	**HSSFeasible** Feasibility of _OperationHSS
static Tool	**HSSTool** High Speed Steel Tool
static java.io.File	**LogFile** Log File
static double	**MachiningTimeCARB** Machining Time for Carbide Operation
static double	**MachiningTimeHSS** Machining Time for High Speed Steel Operation
static boolean	**MinimumTimeCondition** Checks Minimum Processing Time Condition
static boolean	**NextOperationExists** Checks if a new Batch is available
static int	**numberCARBTools** Number of Carbide Tools
static int	**numberHSSTools** Number of High Speed Steel Tools
static int	**ToolSelected** Operation Selected HSS/Carbide

Constructor Summary

INTELLIDRILL()

Method Summary

static void	**accountToolChangeOverTime**() Estimates the Tool Change Over Time and
static void	**adaptiveDecision**() Executes the Adaptive Decison Making Module
static void	**checkInitialOperationFeasibility**() Checks Operational Feasibility
static void	**checkOperationFeasibility**() Checks if Operation Factor of Safety is greater than the Threshold Factor of Safety
static void	**checkOperationToolAvailability**() Checks the Availability of Tray for the Operation
static boolean	**continueDecisionMaking**() Checks if Final Decision is Taken
static java.lang.String	**DecisionStatus**() Prints the Decision Status to a String
static int	**estimateInitialToolRequirement**() Estimates the Initial Tool Requirement for the Operation
static void	**estimateMinimumProcessingCondition**() Estimates the Machining Parameters for the Minimum Processing Time condition
static int	**estimateToolRequirement**() Estimates the Tool requirement for the Operation
static void	**finalizeSystem**() Finalizes the System

static void	**initializeSystem**() Initializes the Decision Support System
static void	**LogDecisionMaking**() Logs Decision Making Process
static void	**LogDecisionMaking**(java.lang.String LogData) Logs the Decision Making Process with specified value.
static void	**printDecision**() Prints the Decision taken
static void	**printOperationSetup**() Prints the Operation Setup
static void	**setOperationParameters**() Sets the Operation Parameters
static void	**setOperationParameters**(double _BatchTime) Sets Operation Parameters with Initial Batch Processing Time
static int	**ToolCostAnalysis**(int _numberHSS, int _numberCARB) Tool Cost Analysis
static boolean	**ToolDecision**(Operation _Operation) Checks Feasibility of Operation and estimates the Cut-off Factor of Safety
static boolean	**ToolDecision**(Operation _Operation, Tool _Tool) Checks Feasibility of Operation with specified Tool and estimates the Cut-off Factor of Safety

Methods inherited from class Debug

print, print, print, print, print, print, print, printTitle

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

_OperationHSS

public static Operation **_OperationHSS**

Operation with High Speed Steel Machining

_OperationCARB

public static Operation **_OperationCARB**

Operation with Carbide Machining

HSSTool

public static Tool **HSSTool**

High Speed Steel Tool

CarbideTool

public static Tool **CarbideTool**

Carbide Tool

HSSFeasible

public static boolean **HSSFeasible**

Feasibility of _OperationHSS

CARBFeasible

public static boolean **CARBFeasible**

feasibility of _OperationCARB

NextOperationExists

public static boolean **NextOperationExists**

Checks if a new Batch is available

ToolSelected

public static int **ToolSelected**

Operation Selected HSS/Carbide

---

numberHSSTools

public static int **numberHSSTools**

Number of High Speed Steel Tools

---

numberCARBTools

public static int **numberCARBTools**

Number of Carbide Tools

MachiningTimeHSS

public static double **MachiningTimeHSS**

Machining Time for High Speed Steel Operation

MachiningTimeCARB

public static double **MachiningTimeCARB**

Machining Time for Carbide Operation

LogFile

public static java.io.File **LogFile**

Log File

---

d

public static Debug **d**

Debug Console

MinimumTimeCondition

public static boolean **MinimumTimeCondition**

Checks Minimum Processing Time Condition

ExceedsLimit

public static boolean **ExceedsLimit**

flags if MaterialRemovalRate is greater than Maximum Allowed Value

Constructor Detail

INTELLIDRILL

public **INTELLIDRILL**()

Method Detail

initializeSystem

public static void **initializeSystem**()

Initializes the Decision Support System

setOperationParameters

public static void **setOperationParameters**()

Sets the Operation Parameters

---

setOperationParameters

public static void **setOperationParameters**(double _BatchTime)

Sets Operation Parameters with Initial Batch Processing TIme

Parameters:

_BatchTime - Initial Batch Time (min)

---

estimateMinimumProcessingCondition

public static void **estimateMinimumProcessingCondition**()

Estimates the Machining Parameters for the Minimum Processing Time condition

---

finalizeSystem

public static void **finalizeSystem**()

Finalizes the System

ToolDecision

public static boolean **ToolDecision**(Operation _Operation,
Tool _Tool)

Checks Feasibility of Operation with specified Tool and estimates the Cut-off Factor of Safety

Parameters:

_Operation - Operation

_Tool - Tool

Returns:

Machining Feasibility

ToolDecision

public static boolean **ToolDecision**(Operation _Operation)

Checks Feasibility of Operation and estimates the Cut-off Factor of Safety

Parameters:

_Operation - Operation

Returns:

Machining Feasibility

---

checkInitialOperationFeasibility

public static void **checkInitialOperationFeasibility**()

Checks Operational Feasibility

---

checkOperationFeasibility

public static void **checkOperationFeasibility**()

Checks if Operation Factor of Safety is greater than the Threshold Factor of Safety

---

checkOperationToolAvailability

public static void **checkOperationToolAvailability**()

Checks the Availability of Tray for the Operation

ToolCostAnalysis

public static int **ToolCostAnalysis**(int _numberHSS,
int _numberCARB)

Tool Cost Analysis

Parameters:

_numberHSS - Number of HSS Tools

_numberCARB - Number of Carbide Tools

Returns:

Tool Selection Index [0-HSS, 1-Carbide]

---

estimateInitialToolRequirement

public static int **estimateInitialToolRequirement**()

Estimates the Initial Tool Requirement for the Operation

Returns:

Tool Selection Index

---

estimateToolRequirement

public static int **estimateToolRequirement**()

Estimates the Tool requirement for the Operation

Returns:

Tool Selection Index

---

accountToolChangeOverTime

public static void **accountToolChangeOverTime**()

Estimates the Tool Change Over Time and

---

continueDecisionMaking

public static boolean **continueDecisionMaking**()

Checks if Final Decision is Taken

Returns:

Decison Making Complete

---

printOperationSetup

public static void **printOperationSetup**()

Prints the Operation Setup

DecisionStatus

public static java.lang.String **DecisionStatus**()

Prints the Decision Status to a String

Returns:

Decision Status

---

printDecision

public static void **printDecision**()

Prints the Decision taken

LogDecisionMaking

public static void **LogDecisionMaking**(java.lang.String LogData)

Logs the Decision Making Process with specified value

Parameters:

LogData - Log Entry Data

LogDecisionMaking

public static void **LogDecisionMaking**()

Logs Decision Making Process

adaptiveDecision

public static void **adaptiveDecision**()

Executes the Adaptive Decison Making Module

VITA

Karthik Chander was born on 29th April, 1979 in Bangalore, India. He completed his Bachelor of Engineering program at Bangalore University, Bangalore, India in the Department of Industrial Engineering and Management in October 2000. In August 2001, he enrolled in the Master of Science program at Texas A&M University in the Department of Industrial Engineering.

Permanent Address:
1270, "Sree Rama Nilaya",
7th cross, J.P.Nagar 1st Phase,
Bangalore, Karnataka- 560078
India.